FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

# Design of Systems Supporting Compliance Management

by

*Marek Skotnica*

A dissertation thesis submitted to
the Faculty of Information Technology, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of Doctor.

Doctoral study programme: Informatics
Department of Software Engineering

Prague, January 2023

**Supervisor:**
doc. Robert Pergl, Ph.D.
Department of Theoretical Computer Science
Department of Software Engineering
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

# Abstract

Compliance management is understood as complying with internal or external regulations in a company. To comply with legal regulations alone, it was estimated that the companies spend 6.2% of their turnover. Therefore, digitalization of compliance management efforts is desirable, and a discipline of enterprise engineering (EE) can improve state of the art. Furthermore, a novel field of decentralized compliance management based on blockchain technology is emerging and needs deeper exploration.

This dissertation thesis provides an overview of the state-of-the-art approaches to designing compliance management systems, their relation to the EE discipline, and its DEMO methodology. The contributions of this thesis are divided into two areas. The first area is centralized compliance management systems that focuses on finding and bridging the gaps between the EE theories and their software implementation. The second area is decentralized compliance management systems that focuses on exploring possibilities of applying blockchain technology in the compliance domain.

In the first area of the research, we collaborated with an industry partner. We identified the most significant gap as a need for an execution language for DEMO models. The FAR ontology and DEMO machine formalizations were designed and developed to bridge this gap. The industry partner further used the proposed formalizations to develop a software system successfully deployed to their customers. The second most significant contribution was an experiment to apply EE theories to improve the quality of process-based software requirements. The experiment consisted of 32 case studies where 115 276 words of the legal text were analyzed. The evaluation of the case studies revealed missing process steps ($\overline{80.10}\%$) and actor roles.

The second area's main focus was formalizing a domain-specific language DasContract to model decentralized compliance processes and then provide support to generate blockchain smart contracts from the developed formalization. Two complex case studies were created – a decentralized mortgage and decentralized EU elections to validate the proposed concept.

**Keywords:** Compliance, Business Process, Information Systems, Decentralized Compliance, Blockchain, Enterprise Engineering.

# Abstrakt

Pojem řízení souladu s předpisy je chápán jako dodržování interních nebo externích předpisů ve společnosti. Jen pro dodržení právních předpisů se odhaduje, že společnosti vynaloží 6,2% svého obratu. Proto je potřeba tuto oblast digitalizovat a disciplína podnikového inženýrství by mohla pomoct ke zlepšení současného stavu. Navíc se objevuje nová oblast decentralizovaného řízení souladu s předpisy založené na technologii blockchain, která zatím není dostatečně prozkoumaná.

Tato disertační práce poskytuje přehled o nejmodernějších přístupech k navrhování systémů řízení souladu s předpisy, jejich vztahu k disciplíně podnikového inženýrství a metodice DEMO. Přínosy této práce jsou rozděleny do dvou oblastí. První oblastí jsou centralizované systémy řízení souladu s předpisy, kde se práce soustředí na nalezení a překlenutí mezer mezi teoriemi podnikového inženýrství a jejich softwarovou implementací. Druhou oblastí jsou decentralizované systémy řízení souladu s předpisy, kde se práce soustředí na zkoumání možností uplatnění technologie blockchain v doméně řízení souladu s předpisy.

Na první oblasti výzkumu jsme spolupracovali s průmyslovým partnerem. Nejvýznamnější mezeru jsme identifikovali jako potřebu prováděcího jazyka pro DEMO modely. K překlenutí této mezery byly vyvinuty ontologie a formalizace prováděcího jazyka pro DEMO modely. Průmyslový partner využil vyvinutou formalizaci k vývoji softwarového systému úspěšně nasazeného u svých zákazníků. Druhým nejvýznamnějším přínosem této práce byl experiment s aplikací teorií podnikového inženýrství pro zlepšení kvality procesních modelů v softwarových specifikacích. Experiment sestával z 32 případových studií, kde bylo analyzováno 115 276 slov právního textu. Vyhodnocení experimentu odhalilo chybějící procesní kroky ($\overline{80.10}$%) a organizační role.

Druhá oblast práce se zaměřila především na formalizaci a vývoj doménově specifického jazyka DasContract, který vznikl za účelem modelování decentralizovaných procesů dodržování předpisů a následné generování smart kontraktů technologie blockchain. Pro ověření navrženého jazyka byly vytvořeny dvě komplexní případové studie – decentralizovaná hypotéka a decentralizované volby do EU.

**Klíčová slova:** soulad s předpisy, podnikový proces, informační systém, decentralizace, blockchain, podnikové inženýrství

# Acknowledgements

First of all, I am extremely grateful to my dissertation thesis supervisor doc. Ing. Robert Pergl, Ph.D. He provided me with excellent guidance and support in the complex world of science. Despite all the challenges I have encountered, he was always there, with his patience and positive attitude. Additionally, I could not have undertaken this journey without Dr. Steven van Kervel and his ForMetis company. It was a very rewarding experience working with him and people from ForMetis on applying the research in practice.

Through my PhD journey, I have discovered a passion for learning and teaching. And that was mostly thanks to the amazing students I had an opportunity to teach and supervise. Special thanks to Barbora Hornáčková, Martina Lassaková, and Jan Klicpera who became co-authors of the papers that are part of this research.

I am also grateful to my colleagues from CCMi research group, namely Dr. Ondřej Dvořák, Ing. David Šenkýř, and others. I would also like to acknowledge my international co-authors Marta Aparício, and Sérgio Guerreiro.

Special thanks go to the staff of the Department of Software Engineering, who maintained a pleasant and flexible environment for my research. I would like to express special thanks to the department management for providing most of the funding for my research. My research has also been partially supported by the Grant Agency of the Czech Technical University in Prague, grants No. SGS SGS17/120/OHK3/1T/18, SGS18/120/OHK3/1T/18, and SGS20/209/OHK3/3T/18. The research was also partially supported by the ForMetis company.

Lastly, I would be remiss in not mentioning my family for their infinite patience, care, and love.

## Dedication

*To my parents.*

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Abbreviations

### Compliance Abbreviations

| | |
|---|---|
| **CobiT** | Control Objectives for Information and Related Technologies |
| **EU** | European Union |
| **GDPR** | General Data Protection Regulation |
| **GRC** | Governance, Risk, and Compliance |
| **HIPAA** | Health Insurance Portability and Accountability Act |
| **ISO** | International Organization for Standardization |
| **ITIL** | Information Technology Infrastructure Library |
| **PCI DSS** | Payment Card Industry Data Security Standard |
| **REA** | Resource-Event-Agent |
| **SOX** | Sarbanes–Oxley Act |

### Blockchain Abbreviations

| | |
|---|---|
| **DAC** | Decentralized Autonomous Corporation |
| **DAO** | Decentralized Autonomous Organization |
| **Dapp** | Decentralized Application |
| **DeFi** | Decentralized Finance |
| **DiD** | Decentralized Identity |
| **DS** | Design Science |
| **ERC** | Ethereum Improvement Proposals |
| **SC** | Smart Contract |

### Miscellaneous Abbreviations

| | |
|---|---|
| **API** | Application Programming Interface |
| **BC** | Blockchain |
| **BPEL** | Business Process Execution Language |
| **BPM** | Business Process Management |
| **BPMN** | Business Process Model and Notation |
| **BPMS** | Business Process Management System |
| **BPT** | Business Process Technology |
| **CCO** | Chief Commercial Officer |
| **CIO** | Chief Information Officer |
| **CMMN** | Case Management Model and Notation |
| **CMS** | Content Management System |
| **DSL** | Domain Specific Language |
| **DSR** | Design Science Research |
| **EBNF** | Extended Backus-Naur Form |
| **EPC** | Event-driven Process Chain |
| **GPS** | Global Positioning System |
| **IDE** | Integrated Development Environment |
| **IS** | Information System |
| **IT** | Information Technology |
| **MDD** | Model-Driven Development |
| **MDE** | Model-Driven Engineering |
| **MIT** | Massachusetts Institute of Technology |
| **MVVM** | Model–View–ViewModel |
| **NLP** | Natural language processing |
| **OMG** | Object Management Group |
| **OrgML** | Organization Markup Language |
| **OT** | Operational Technology |
| **REST** | Representational State Transfer |
| **SOAP** | Simple Object Access Protocol |
| **SPA** | Single-Page Application |
| **SW** | Software |
| **TIL** | Transaction Instance Linking |
| **UI** | User Interface |
| **UML** | Unified Modeling Language |
| **USD** | United States Dollar |
| **UX** | User Experience |
| **YAWL** | Yet Another Workflow Language |

### Enterprise Engineering Discipline Abbreviations

| | |
|---|---|
| **AM** | Action Model |
| **C4E** | Comprehensive, Coherent, Concise, Consistent, and Essential |
| **cAct** | Coordination Act |
| **CM** | Construction Model |
| **DELTA** | Discrete Event in Linear Time Automaton |
| **DEMO** | Design and Engineering Methodology for Organizations |
| **DEMOSL** | DEMO Specification Language |
| **EE** | Enterprise Engineering |
| **EET** | EE theories |
| **EEWC** | Enterprise Engineering Working Conference |
| **EIS** | Enterprise Information System |
| **EO** | Enterprise Ontology |
| **FAR** | Fact, Agenda, and Rule |
| **FM** | Fact Model |
| **OCD** | Organization Construction Diagram |
| **OER** | Organization Essence Revealing |
| **PM** | Process Model |
| **PSD** | Process Diagram |
| **PSD** | Process Structure Diagram |
| **PSI** | Performance in Social Interactions |

# Part I

# Introduction

Chapter 1

# Research Overview

The rapid evolution and adoption of information technologies and globalization brought unprecedented opportunities for entrepreneurs from all around the globe. An example of this is the Amazon company, founded in 1994, and over the years, it became a synonym for online shopping. For an Amazon customer, the experience of ordering a book is very convenient, but what goes on behind the scenes is a whole different story. A holistic network of top-notch technologies supports Amazon's operations. Hundreds of parties collaborating, distributed IT systems, sci-fi-looking automated robotic warehouses, and job positions that did not even exist when the company was started.

Furthermore, because Amazon is a publicly-traded multinational company, this sophisticated network must comply with ever-changing legislative, industry standards, and regulations. It is also a subject of various improvement initiatives and audits. Dealing with such issues is a significant problem for all organizations and governments. To comply with legal regulations alone, it was estimated that the companies spend 6.2% of their turnover [120].

The compliance management we will refer to in this thesis is understood as complying with internal or external regulations in a company. Internal compliance is understood as complying with internal company policies such as business processes, business rules, responsibilities for carrying out the work, and organization roles. The scope is further narrowed to the concepts surrounding business processes, their definition, execution, automation, and evaluation of whether the operation of companies is executed as wanted. External compliance is understood as complying with legislative regulations, industry standards (such as ISO standards [82], BASEL [8], or General Data Protection Regulation (GDPR) [46]), and requirements of subjects cooperating with our enterprise. However, we are not experts in any particular compliance domain. Our scope is to help translate the expert domain knowledge and standards into systems that support their execution.

Business process models are blueprints of the organizations. If modeled correctly, they provide a map of the business operation and help people to build the required infrastructure for their support. Nevertheless, if we imagine a blueprint for a company like Amazon, it is incredibly complex, and by the time someone finishes modeling it, it will already

be obsolete. This hints that we cannot apply traditional engineering techniques to design enterprises on a larger scale. We would need something more advanced for this task, something like Google Maps does with the traffic conditions. It shows us live traffic information and adjusts our GPS navigation accordingly to reach the destination in the shortest amount of time. We need a live overview of the operation of an enterprise and be able to adapt to the ever-changing environment and requirements that surround it. However, although some software companies offer products that can support these processes, they could be better. Most companies still use barely-updated word documents to describe the processes and do ad-hoc Excel analysis to evaluate them. The cost and efforts associated with introducing such systems usually outweigh the benefits. More advanced approaches are needed to enable compliance automation.

There are many aspects to solving compliance problems, but blockchain technology has been the most interesting in recent years. The blockchain is mainly known for its digital currency use-cases such as Bitcoin [108], but they are the least interesting ones for our domain. Blockchain technology also enables smart contracts, decentralized identity, decentralized autonomous organizations, and the realization of a concept called decentralized compliance.

A report The future of compliance by a consultation company KPMG distinguishes three types of compliance: "*Centralized*: the compliance function retains direct control over all compliance-related activities and execution of controls. A common structure in highly regulated sectors such as financial services. It often involves a large team of dedicated compliance officers. *Decentralized*: Compliance is embedded in existing functions (finance / human resources etc.). Compliance activities are carried out by function, with limited central oversight. *Hybrid*: Responsibilities for compliance activities are delegated within the organization, but the oversight and ultimate responsibility are borne centrally or regionally (if the corporation is a large multinational)." [88] The KPMG report also outlines the trend of "the transformation of the compliance role: from highly centralized to a hybrid or even a decentralized system" [88]. This trend towards decentralization in compliance is further supported by the Europeans Commission's blockchain strategy of building its pan-European public services blockchain [45].

From an individual's perspective, decentralized compliance may provide a paradigm shift in certain areas and make some traditional compliance requirements obsolete. We will explain this paradigm using the process of taking a mortgage as an example. Currently, a central authority such as a bank is required to oversee the process of taking a mortgage. The bank must follow many compliance requirements that result in high costs for the customer. However, when the mortgage process is decentralized, it is possible to write all parameters, rules, and steps into a smart contract and upload it to the blockchain, ensuring that the mortgage is executed between the parties. A bank can be made obsolete in this case, and because the contract is entirely digitized, the compliance requirements can be fully automated. This approach may significantly reduce costs for the end customers of the mortgage process.

Practical use cases of the decentralized compliance approach currently running in an European Union (EU) blockchain are university applications, diploma verifications, and

EU grant applications [41]. Another example is a decentralized validation of tax processes proposed in [48]. However, all use cases for decentralized compliance are yet to be discovered as the full adoption of blockchain technology is expected in 2030, according to Gartner [54].

This chapter provides an overview of our research on the design of software systems supporting compliance management. The research domain and its challenges are introduced and narrowed in Section 1.1 and Section 1.2. The research problem is formulated in Section 1.3. The research questions and objectives are set in Section 1.4. The research design is briefly introduced inSection 1.5. The main contributions are summarized in Section 1.6. The thesis structure is introduced in the Section 1.7. Finally, this chapter is summarized in Section 1.8.

## 1.1 Challenges of Centralized Compliance

According to the KPMG report Innovating compliance through automation, only one in five enterprises has a well-defined enterprise-wide strategy to automate compliance [89]. The report also states that according to the chief commercial officers (CIOs) and chief information officers (CCOs) survey, the top challenges to implementing compliance automation as: dependencies were misunderstood and/or insufficiently managed, attention from leadership and/or stakeholders, metrics for measuring progress were insufficient, resources to support the automation were unavailable, and data was unavailable or did not have the anticipated integrity [89].

Further, the top five compliance challenges were identified in [131] as: 1) Siloed functions – Companies only react to single compliance events and therefore disregard a holistic view of the overall compliance management in an organization. 2) Disconnected systems – The compliance is managed across multiple business lines, functions, and locations, therefore missing ways to exchange and consolidate data. 3) Manual processes – When every compliance requirement must be processed manually, it leads to tremendous work and opens the door to human errors. 4) Incomplete or nonexistent metrics – When compliance requirements are fulfilled through lengthy manual processing, compliance reports are focused on the past rather than the future. 5) No visibility – Without a holistic view on the compliance-related activities, it is nearly impossible to identify inconsistencies and gaps in compliance management. This may result in some compliance issues not being addressed.

In scientific literature, the article [151] states that although the compliance management is usually supported by some IT systems, the systems are inadequate and do not fully address the needs of organizations. Moreover, according to the article, academic research efforts do not align with the needs of organizations. The factors contributing to inadequate compliance management solutions are: 1) Lack of holistic practices. 2) Lack of IT support/tools. 3) Lack of compliance knowledge base. [151] The inefficiency of existing solutions is further supported in [20]: "Research suggests that Content Management Systems (CMSs) may in some cases live up to their promise, at least to some degree, but

so far the available studies also suggest that the impacts of CMSs in reducing regulatory violations may be rather modest.“.

An extensive survey consisting of 79 papers [71] concludes that: ”None of the surveyed frameworks fully cover all dimensions of the compliance problem as a number of challenges still remain“. The most important challenges in [71] were identified as: 1) ”the need to develop automated techniques for extracting norms from legal documents, and formal languages expressive enough to model various types of legal norms, and norms pertaining data and resources“. 2)”there is need for fully automated techniques that are scalable and computationally efficient“ 3)”there is a huge room for investigation on the issue related to handling maintainability of ever-changing business processes and legal norms.“

A report from Capgemini [18] suggests that the compliance management challenges should be tackled via BPM. The report states that using BPM will help improve the processes and make it easier to respond to regulatory changes as they come. An interesting approach to BPM that could help with compliance-related challenges is presented by the discipline of Enterprise Engineering (EE) [31]. The EE aims to assist professionals in mastering the complexity of modern enterprises by providing suitable theories and methodologies. The discipline introduces Design and Engineering Methodology for Organizations (DEMO) [32] based on its formal theories. This methodology introduces a way of modeling that consists of the DEMO specification language and four visual aspect models – Cooperation Model, Action Model, Process Model, and Fact Model. The DEMO methodology provides many benefits over state-of-the-art modeling techniques. However, there are known gaps between the description of a social system and its information technology support. The key missing knowledge is the execution of the models and their implementation in compliance management systems, which are both out-of-scope of the original theories.

The last issue we observed in the academic literature was the need for real-world case studies. This might be because the implemented compliance management processes and practices are not publicly available because they are the property of the companies. In our professional experience, signing a non-disclosure agreement before working with company processes is common practice. The case studies that are available in the academic literature are usually simple examples that usually don't really reflect the scope and size of the real environment.

## 1.2 Challenges of Decentralized Compliance

The decentralized way of handling compliance was proposed as a possible future of compliance in KPMG report [88]. However, more profound guidance on implementing decentralized compliance was not provided. The lack of real-world manifestations of blockchain technology seems to be prevalent. In [134], the article discusses many potential decentralization use cases, however concludes with: “Nevertheless, being relatively infant technology, many concerns such as security, privacy, efficiency, scalability, energy consumption, interoperability, regulatory concerns, etc. are yet to be deeply investigated for its overall adoption.”.

The potential use cases for decentralization in various industries are well documented. For example, in healthcare, the article [171] outlines possible use cases in healthcare such as prescription tracking, patient digital identity, cancer registry sharing, and more. The article concludes by outlining healthcare challenges: "the complexities associated with healthcare involvement and regulations create additional challenges inevitably facing blockchain-based systems, such as system evolvability, information privacy, and communication scalability." [171] and invites further research in the space.

Another domain where decentralization is expected to provide the most significant benefit is finance. According to the article [133], decentralization in finance provides a trustless version of traditional financial instruments. Moreover, it creates new financial instruments such as atomic swaps, liquidity pools, decentralized stablecoins, and flash loans. The article concludes with: "While this technology has great potential, there are certain risks involved. Smart contracts can have security issues that may allow for unintended usage, and scalability issues limit the number of users." [133] The mentioned risks are manifested in real-world decentralized finance systems and are estimated to result in losses of 3 billion dollars in 2022, according to Bloomberg [2].

## 1.3 Research Problem

The challenges for centralized and decentralized compliance outlined in the previous sections are very broad and impossible to achieve in the scope of a single PhD thesis. In this section, we narrow the scope down to an achievable research problem.

With an increase in compliance management activities, it is hard to gain a holistic view of an enterprise to implement, enforce and change compliance requirements across software systems. This results in increased manual labor, risk of penalties for not being compliant with legal regulations, decreased product quality and decreased ability to evolve an enterprise's core business. The EE is a promising area that could help with the outlined challenges. Although some work is done on applying EE theories and DEMO in BPM practice, DEMO is formulated as implementation-independent. Therefore, further research is required in the application of EE theories to the compliance management domain and process-based IT systems.

The decentralized compliance systems could also benefit from applying BPM practices and EE theories. The improvements in centralized compliance are expected to carry over to the decentralized one. The main challenge in decentralized compliance is the need for more adoption and guidance on implementing such systems safely and methodically.

The common challenge in both centralized and decentralized compliance domains seems to be a lack of real-world case studies. In the case of centralized compliance, the lack of published case studies is due to company privacy. In the case of decentralized compliance, the case studies only exist on a theoretical level. However, the predictions need more technical detail to make them a reality.

To summarize, we narrow our research to exploring how EE theories can improve state-of-the-art centralized and decentralized compliance management systems. The focus will be

on implementing compliance-related business processes with centralized and decentralized IT systems.

## 1.4 Research Objective and Research Questions

The research objective is laid out in this section, and the research questions are formulated.

Based on the research problem defined in the previous section, the research objective is to investigate the state of the art of the EE theories and DEMO methodology, centralized and decentralized compliance systems, software systems supporting BPM, and blockchain technology. Investigate the gaps between the DEMO methodology and its application in the design of centralized and decentralized compliance management systems. For some of the identified gaps, propose artifacts that could contribute to filling them. Finally, provide case studies of the proposed approaches that resemble real-world use cases.

Two main research questions are defined based on the formulated research objective to explore centralized and decentralized aspects of compliance management systems design. Both main research questions have two sub-research questions designed to fill the gaps in the current state-of-the-art as described in the previous sections.

**Research Question 1.** How to design software systems to support business process management requirements based on EE theories and DEMO methodology?

**Sub-research Question 1.** How to support the execution of DEMO models in software systems using an execution language?

**Sub-research Question 2.** How should DEMO models be transformed into Business Process Model and Notation (BPMN) models?

**Research Question 2.** How to digitize business processes using blockchain smart contracts in a methodical way and eliminate programming errors while avoiding a dependency on a particular blockchain implementation?

**Sub-research Question 3.** How can blockchain smart contracts be used in the implementation of a software system based on DEMO methodology?

**Sub-research Question 4.** Is it feasible to generate blockchain smart contracts from a high-level modeling language in an automated methodological way?

## 1.5 Research Design

In our interdisciplinary research, we use the Design Science Research (DSR) [74, 169] strategy to work on the research objective and research questions. In this section, the research design is summarized and discussed in detail later in Chapter 3.

The DSR research strategy of this thesis is outlined in Section 1.5. First, an extensive literature review is performed to identify the gaps in existing knowledge and prevent discovering what was already done. Building on top of existing knowledge is also crucial to maintain the scope of this research.

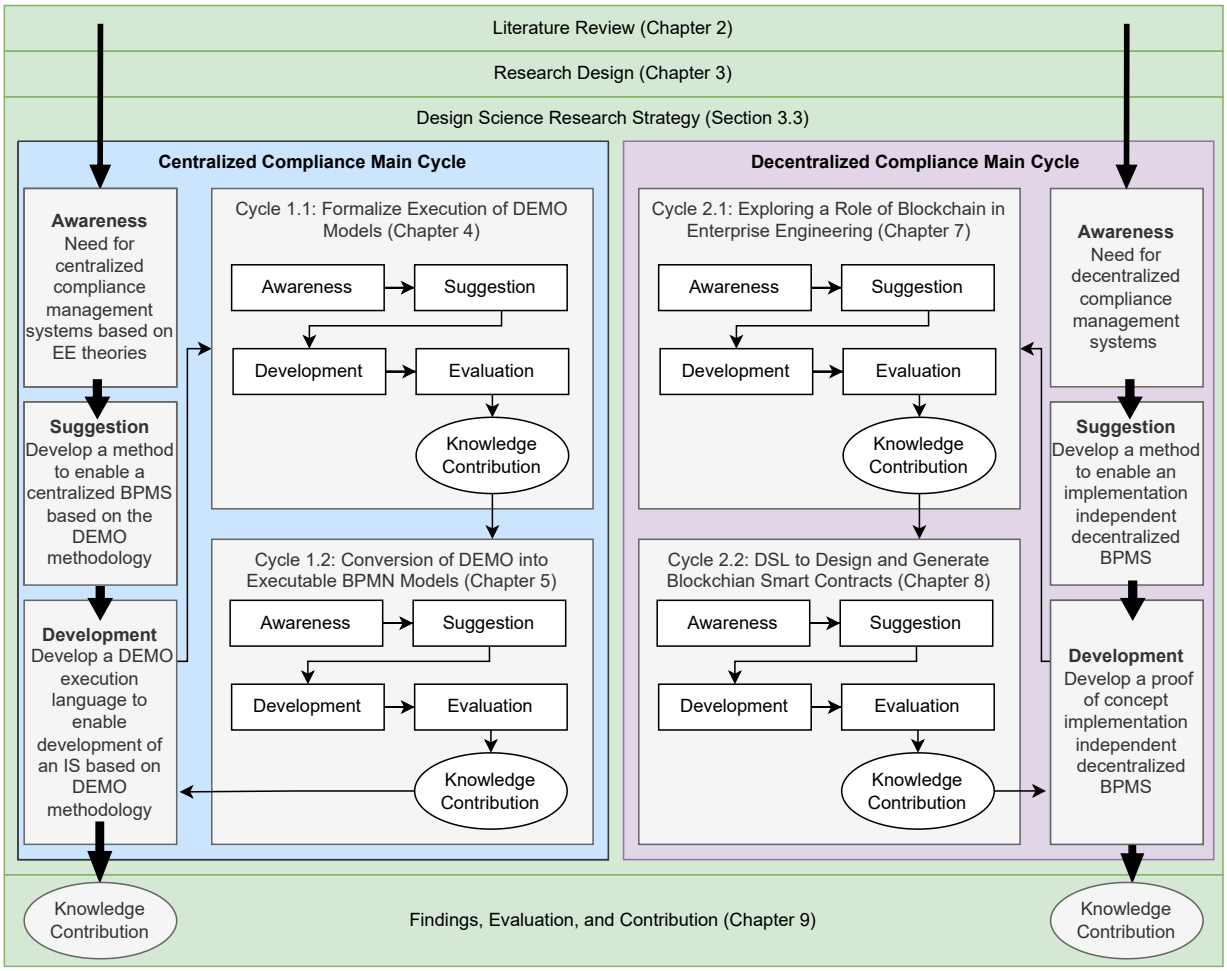


Figure 1.1: DSR research strategy

The DSR strategy used in this thesis consists of several research cycles. After each research cycle, a knowledge contribution is made. This allows adjustment of the research according to the findings from each finished cycle. The DSR approach also allows the utilization of different research techniques. As outlined in Section 1.5 our research strategy consists of two main research cycles.

The first main research cycle concerns centralized compliance management (visualized in light blue) and contains two sub-cycles. The first sub-cycle is focused on a formalization of the execution of DEMO models. The second sub-cycle focuses on converting DEMO models into executable BPMN models.

The second main research cycle concerns decentralized compliance management (vi-

sualized in light purple). The first decentralized sub-research cycle explores the role of blockchain technology in the EE discipline. The second decentralized sub-research cycle is focused on creating a domain specific language (DSL) to design and generate blockchain smart contracts.

The table Table 1.1 provides a mapping of the research cycles to the research questions formulated in Section 1.4.

Table 1.1: Mapping of research questions and DSR strategy

| Research Question | Centralized Cycle | Cycle 1.1 Chapter 4 | Cycle 1.2 Chapter 5 | Decentralized Cycle | Cycle 2.1 Chapter 7 | Cycle 2.2 Chapter 8 |
|---|---|---|---|---|---|---|
| RQ 1 | X | | | | | |
| SRQ 1 | X | X | | | | |
| SRQ 2 | X | | X | | | |
| RQ 2 | | | | X | | |
| SRQ 3 | | | | | X | |
| SRQ 4 | | | | X | | X |

## 1.6 Contributions

This section provides an overview of the most significant contribution of the thesis. The contributions consist of research artifacts and publications. The research artifacts are further narrowed into four categories - formalizations, methods, experiments, and case studies. The artifacts are outlined in Section 9.1.1. The publications are of two categories, the first is the seven peer-reviewed papers published/co-authored by the author, and the second is 22 student theses supervised by the author that contributed to the overall research.

A summary of the most important contributions:

- The research contributed to the publication of 7 peer-reviewed papers with 49 citations (Section 9.1.2). The main author supervised 22 student theses relevant to this thesis (Section 9.1.3), and 3 received the dean's award.
- A formalization of execution of DEMO aspect models called the DEMO Machine (Chapter 4) that allowed:
  - An implementation of a DEMO-based BPM system in a professional company (Section 9.3).
  - Development of a method that converts DEMO models into executable BPMN models (Chapter 5).
  - The FAR ontology was used to allow modeling of REA compliant accounting systems in [78].

- Conducted an experiment to improve the quality of software specifications for compliance management systems (Chapter 6) in the domain of procedural law.
  - The experiment consisted of 32 case studies where 115 276 words of the legal text were analyzed in approximately 2 440 hours.
  - The evaluation of the case studies revealed missing process steps ($\overline{80.10}\%$) and actor roles.
- A domain-specific language DasContract to model blockchain smart contracts was formalized.
  - A method to convert the DasContract models into executable blockchain smart contract code was described (Section 8.3).
  - An open-source proof-of-concept implementation of the DasContract model editor and an algorithm to generate executable blockchain smart contracts was created (Section 8.2.2).
  - Two case studies were created and modeled in the DasContract language based on the possible real-world utility of blockchain.

## 1.7 Structure of the Dissertation Thesis

The thesis structure follows the research design strategy outlined in Section 1.5. The chapter map is shown in Section 1.7. The thesis consists of seven parts:

1. *Introduction*: Describes a motivation behind our research together with formulating the research problem, research objective, and research question. There is also a list of the most important contributions of this thesis.
2. *Literature Review*: Provides an overview of existing literature in the context of our research.
3. *Our Approach*: Provides an overview of our research design used to answer the research questions.
4. *Centralized Compliance Management*: Presents our contributions related to centralized compliance management.
5. *Decentralized Compliance Management*: Presents our contributions related to decentralized compliance management.
6. *Conclusions*: Summarizes the results of our research, evaluates the research questions, and presents our research contributions. Lastly, it suggests possible topics for further research and concludes the thesis.

7. *Publications*: Contains the cited sources used in this thesis. Provides a list of the reviewed publications of the author relevant to the thesis and a list of the relevant supervised theses.

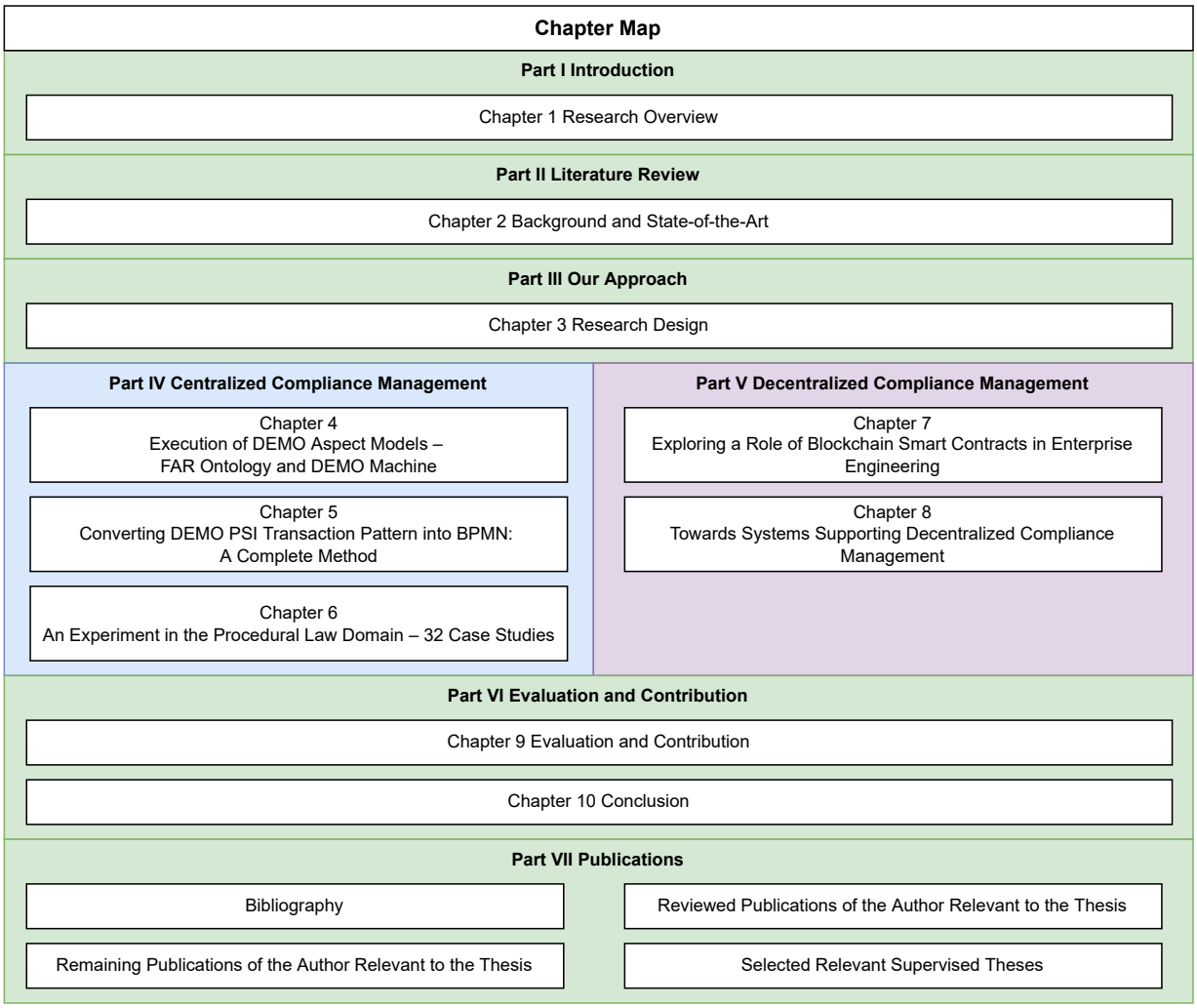


Figure 1.2: Chapter map of the thesis

## 1.8 Chapter Summary

In this chapter, we introduced the main challenges of centralized and decentralized compliance management. The scope of the research was narrowed to an achievable research problem and research objective. Research questions were formulated, and a research strategy was outlined to answer them. Furthermore, the assumptions, scope, and limitations were discussed. A summary of the main contributions was provided. Moreover, the structure of the thesis is outlined.

# Part II

# Literature Review

Chapter 2

# Background and State of the Art

This chapter introduces a theoretical background, state-of-the-art, and related research. Compliance management is an extensive area; therefore, this chapter summarizes only the most essential sources and points to other sources for further details.

In Section 2.1, compliance management is introduced, and a high-level overview of compliance management strategies is presented. An introduction to blockchain technology relevant to decentralized compliance management can also be found in Section 2.1. Later, approaches to modeling and execution of business processes based on the compliance rules are summarized in Section 2.2. The enterprise engineering discipline is introduced in Section 2.3 as an application of the EE theories and methodologies to the domain of compliance management is the main focus of our research. The section also includes a guide on how to read the DEMO models (Section 2.3.5) to make this thesis more accessible for an audience not familiar with EE.

## 2.1 Compliance Management

Although our work is focused only on compliance management, the term compliance is usually used together with risk and governance management. GRC management is a broad topic. We do not focus on being experts in any particular domain; instead, we aim to help the domain experts translate their requirements into high-quality models that can be later executed in software systems. Our supervised work by M. Mužák [A.26] provides a more profound overview of the GRC topic.

**Governance** has almost as many definitions as people talking about it. It generally means who is responsible for what. In practice, it is a sensitive and political topic. For our purposes of supporting the governance with IT, we choose this one: "IT governance is the organizational capacity exercised by the board, executive management and IT management to control the formulation and implementation of IT strategy and ensure the fusion of business and IT."[163] The governance in IT is usually implemented in companies by following a set of standards such as Information Technology Infrastructure Library (ITIL) [22], The CobiT Framework [64], and ISO 27001 [81].

**Risk** is a susceptible area in organizations because it is about managing the damage when something goes wrong. Risk management usually balances between carefully identifying and preventing the risks on one side and evaluating whether the cost of identifying and preventing the risk is no higher than the damage caused by letting the dangerous situation happen. For our work related to IT support of risk management systems, we picked the following definitions: "There are three fundamental qualities of information which are vulnerable to risk and which, therefore, need to be protected at all times, namely availability, integrity, and confidentiality."[65] "The risks that threaten the security of its information and computer resources need to be assessed and managed in a proper way, and the necessary security controls need to be implemented and managed effectively."[7] The most common standard to implement risk management in IT is ISO 31000 [79].

**Compliance** generally means fulfilling the organization's values, policies, and procedures. They can be either external – legal requirements, industry standards, customer rights, or internal – stakeholder requirements, governance, risk policies, organization strategy, processes, and know-how on delivering products to the customer. Our work focuses only on the procedural part of compliance and its support by software systems. We adopt our definition of compliance from Gartner: "The process of adhering to policies and decisions. Policies can be derived from internal directives, procedures, requirements, or external laws, regulations, standards, and agreements." [55]. Examples of external compliance requirements are: BASEL [8] and The Sarbanes-Oxley Act (SOX) [3] – financial regulations for banks, GDPR [46] – a regulation to protect the personal information of EU citizens, HIPAA [75] – American healthcare regulations, PCI DSS [123] – regulations for merchants accepting credit cards. The internal company processes are usually hard to get by as they are a part of the organization's know-how and are protected by non-disclosure agreements.

The maturity of GRC management in organizations can be measured by the GRC Capability Model that consists of 4 integrated components [104]:

- Learn: Examine and analyze context, culture, and stakeholders to learn what the organization needs to know to establish and support objectives and strategies.
- Align: Align performance, risk and compliance objectives, strategies, decision-making criteria, actions, and controls with context, culture, and stakeholder requirements.
- Perform: Address threats, opportunities, and requirements by encouraging desired conduct and events and preventing what is undesired through applying proactive, detective, and responsive actions and controls.
- Review: Conduct activities to monitor and improve the design and operating effectiveness of all actions and controls, including their continued alignment with objectives and strategies.

### 2.1.1 Compliance Management Strategies

In [71], the authors provide an extensive overview of the state-of-the-art business process compliance and a schematic overview of compliance management strategies. The schema

is presented in Figure 2.1 and we cite it's description of the strategies [71]:

- **Design-time** (otherwise, pre-execution time) is a preventive compliance management strategy where business processes are assessed for any non-compliant patterns at the very early stages of the process design. As such, in this approach, the compliance requirements are captured through a logic-based requirements modeling framework and propagated into business processes. Any non-compliant issues can be detected in the very early stages, thus saving an enterprise's efforts, time, and financial resources.

- **Run-time** (otherwise, execution-time) compliance checking is a strategy by which enterprises use specialized software products that produce compliance reports while the processes are being executed. This approach has a limited scope because it still falls in the after-the-fact category. Also, it requires human intervention to rectify the detected problems.

- **Auditing** (otherwise, post-execution) is a strategy by which specialized compliance consultants manually analyze the logs generated by the processes to detect possible violations. The main drawback of this strategy is the use of manual checks, which requires a great deal of time and resources, and the use of manual checks is thus a costly venture. However, the increased pressure and threat of possible criminal prosecutions make the auditing method a less attractive compliance reporting strategy.

Our research is focused on applying the EE theories to the design-time process engineering and then enabling the creation of an information system based on compliant business process models. The used approaches will be further elaborated in Section 2.2 and Section 2.3.

### 2.1.2 Business Process Management

Gartner IT Glossary defines BPM as: "A discipline that uses various methods to discover, model, analyze, measure, improve and optimize business processes. A business process coordinates people's behavior, systems, information, and things to produce business outcomes in support of a business strategy. Processes can be structured and repeatable or unstructured and variable. Though not required, technologies are often used with BPM. BPM is key to aligning IT/OT investments to business strategy." [55]. We do also include an ambitious definition: "Business Process Management (BPM) is a comprehensive system for managing and transforming organizational operations, based on what is arguably the first set of new ideas about organizational performance since the Industrial Revolution." [68]

The BPM usually consists of parts shown in Figure 2.2:

1. Design – Identifying existing processes and defining new ones

2. Model – Modeling the theoretical design using some tool. It can be a BPMS as well as a paper document.

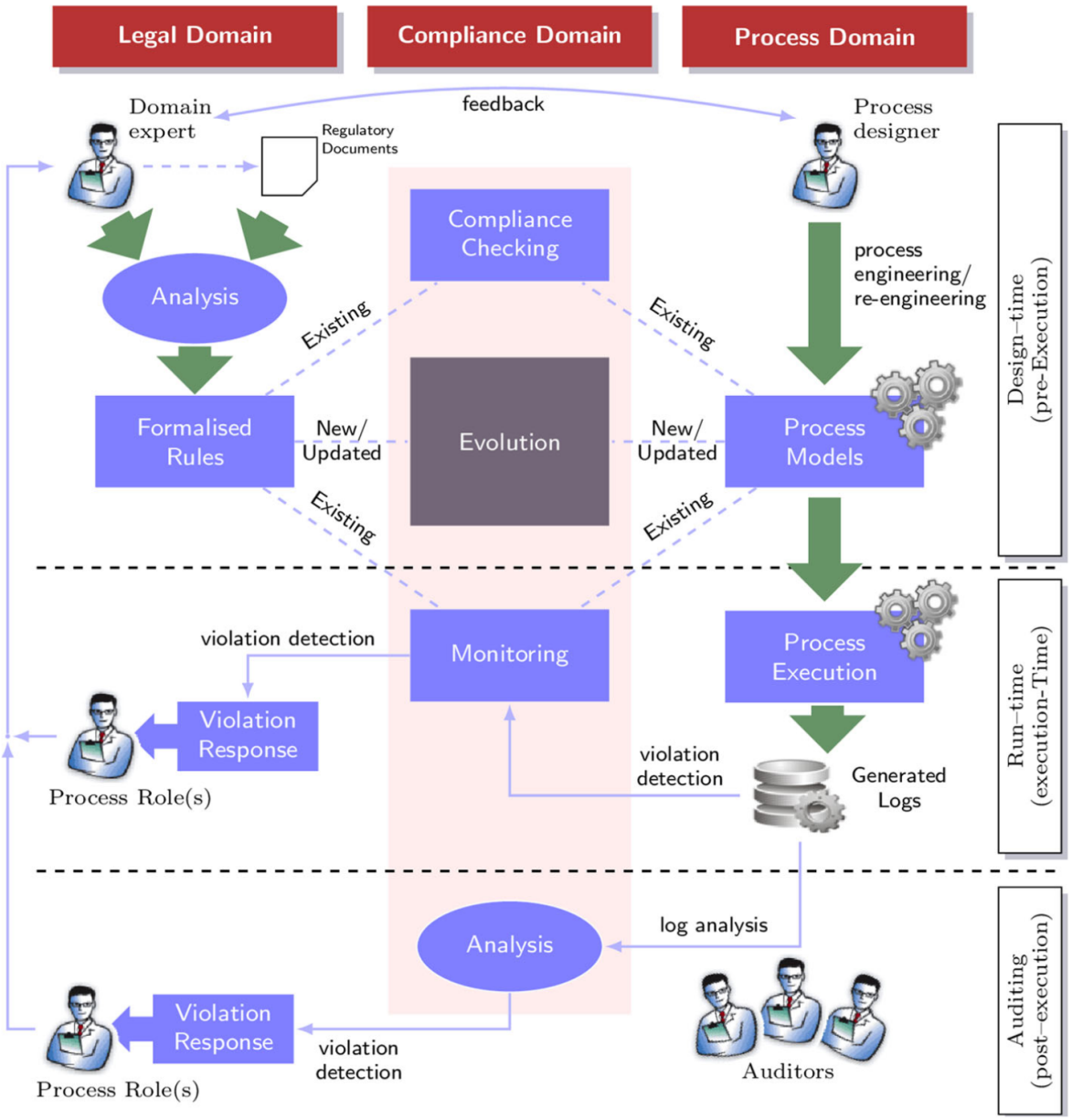


Figure 2.1: Compliance management strategies [71]

3. Execute – Put the process into the production environment so people can start acting according to it.
4. Monitor – Monitor metrics that are important for the process.
5. Optimize – Improve the process using the metrics captured in the previous step.
6. Re-engineering – Sometimes, the solution becomes so complex that it is better to re-engineer the whole process.

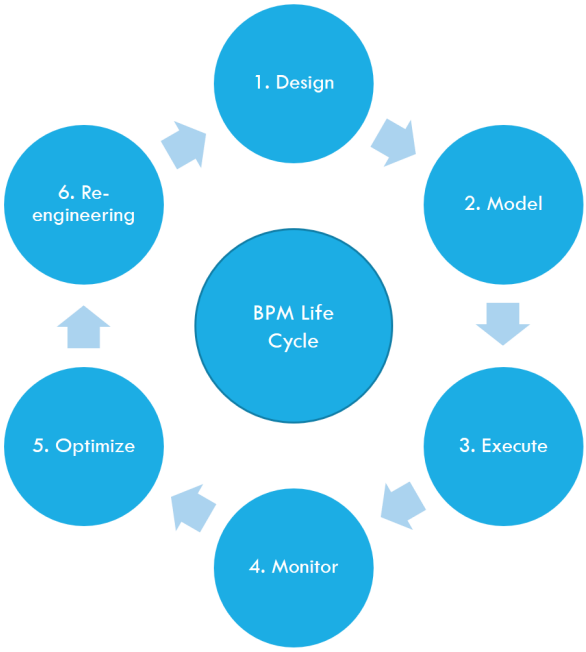


Figure 2.2: BPM life-cycle

### 2.1.3 Decentralized Compliance and Blockchain

Blockchain (BC) is a technology popularized in a whitepaper [108] by Satoshi Nakamoto[1] It is mainly known for its use with Bitcoin as it is its underlying technology. It is a new way of looking at transactions, assets exchange, or even whole organizations. It introduces a decentralized, autonomous, replicated, and secure database. Based on cryptography offers a trustless [126] network with no need for an intermediary, resulting in significant resource and also time-saving. The possibilities for applying this technology are extensive, and it could be effectively used in most parts of our world.

The public blockchains' primary use is the decentralized finance (DeFi) used for decentralized exchanges and crypto loans. However, there are many problems DeFi is currently facing. The first one is network congestion, which increases the transaction processing time and price (up to 12 USD per transaction [170]). Another issue is the smart contract programming errors. In one of the many incidents, an input error led to a token price plunging 25% [110]. Furthermore, according to the CoinGeco survey [21], only 40% of the DeFi users can read and understand the smart contract's source code.

A part of our research builds on top of blockchain technology and shows how to use it for the design and execution of decentralized compliance systems. The topic was explored in our publications [A.4, 138, A.6], as well as in many of our supervised theses [A.21, A.17, A.27, A.28, A.30, A.29].

[1]Satoshi Nakamoto is probably a pseudonym for either one person or a group of people, the identity is currently unknown.

#### 2.1.3.1 Private and Public Blockchain

The original intention of blockchain and bitcoin-like implementation was to create a public network, but due to some limitations, it brings, private blockchains have been developed as well. The main disadvantage of a public blockchain is the computational power it needs to maintain the ledger when used on a large scale. The second issue is the openness of the system and a consequent lack of privacy of transactions and their content. The difference between public and private blockchain is based on controlling who can be part of the network; in more detail, it means who can participate in the network and in which parts, who can execute the consensus protocol, and manage the ledger. It is also referred to as a permissioned blockchain, in contrast to the public blockchain, which is permissionless. It requires an invitation to join a private blockchain, where the access control mechanism may vary [150]. This means that in a private blockchain, there is control over the extent to which it is decentralized and anonymous [124]. Private blockchains are faster, as there is a reduced number of processing nodes, and the transaction costs might be lower [124]. On the other hand, this access control brings extra costs and complexity to the process of maintaining or joining the blockchain. There are also hybrid solutions combining private and public blockchains referred to as "consortium blockchains" [124].

#### 2.1.3.2 Smart Contracts

The idea of smart contracts (SC) [152] is to offer more complex solutions than only sell and buy transactions. A smart contract is a transaction embedded in blockchain that contains enhanced logic – an executable contract has its own data storage and can access other resources to evaluate its current state and perform actions – a contract made of code. "A smart contract is a set of commitments that are defined in digital form, including the agreement on how contract participants shall fulfill these commitments." [109]

The main characteristic of a programmable smart contract is that it does not require trust between parties. After its creation in blockchain, it would execute itself immutably. The parties would not need to be in further contact or use an intermediary; it would be autonomous instead. Smart contracts are not doing something that was not possible before; however, they reduce the complexity of common problems, and they help with automation [149].

#### 2.1.3.3 Dapps, DAOs, and DACs

Smart contracts have the potential not just to be simple contracts between several parties, but over time they could become very complex systems involving many parties and resources. The definition of decentralized applications (Dapps) can vary, but in general, it refers to open-source autonomous applications that use the decentralized network and executes across decentralized network nodes [27]

When further enhancing Dapps and creating applications that handle complicated functionality, they interconnect. This all happens in an autonomous decentralized manner; we

may create decentralized autonomous organizations (DAOs) and even decentralized autonomous corporations (DACs). DAOs and DACs are "a concept derived from artificial intelligence. Here, a decentralized network of autonomous agents performs tasks, which can be conceived in the model of a corporation running without any human involvement under the control of a set of business rules. In a DAO/DAC, there are smart contracts as agents running on blockchains that execute ranges of pre-specified or pre-approved tasks based on events and changing conditions." [149]

## 2.2 Compliance Management Systems Modeling and Execution

This section provides an overview of languages used for business process modeling in the context of software execution.

Business process modeling languages are divided into three categories according to their monography [160]. The first category is discussed in Section 2.2.1 and involves formal languages based on formal mathematical foundations that exhibit high precision and require high modeling skills. The second category is overviewed in Section 2.2.2 and involves conceptual languages that are modeler friendly but lack ontological qualities. The third category presented in Section 2.2.2 involves execution languages used to execute conceptual models in software solutions. An attempt to bridge formal and conceptual languages appears to be the EE theories presented in Section 2.3. Section 2.3.4 defines the concept of a language's ontological quality.

### 2.2.1 Formal Languages

According to Aalst, formal languages are based on mathematical models [160]. An example of this class is *process calculus*, which is defined algebraically [6]. For complex processes in the context of business process modeling for software execution, Petri nets [128] can be used. In addition to mathematical formalism, they provide a visual representation of underlying formal constructs. Other languages based on Petri nets have been created, such as YAWL [159], which allows for advanced properties, such as complex synchronization between process instances.

Formal properties allow models to be simulated, optimized, checked for inconsistencies or possible deadlocks, etc. For Petri nets, many tools allow such functions [154] and can improve the quality of the software specifications.

However, based on a survey [121] with n=130, formal languages are not used in practice for IT requirements in business process-based systems. Languages are typically used in high-assurance environments, in which process errors may result in large financial or physical damage. An example of a field in which formal languages are applied is manufacturing [1].

### 2.2.2 Conceptual and Execution Languages

According to Aalst, "users in practice often have problems using formal languages due to the rigorous semantics (making it impossible to leave things intentionally vague) and low-level nature" [160]. Therefore, higher-level conceptual languages are used. The most used conceptual languages in practice [121] are BPMN [114], the UML activity diagram [115], and event-driven process chain (EPC) [100].

These are relatively informal, meaning that their semantics are not mathematically defined and do not allow for analysis [160]. Attempts have been made to formalize these languages ex-post; UML activity diagrams can be converted into alloy reasoners [87]. An approach to converting EPC to Petri nets for analysis was proposed in [158]. A BPMN can also be converted into Petri nets [92]. However, because of complexity, only a subset of a conceptual language is typically formalized and does not therefore allow the analysis of complex models. Moreover, some operations, such as reduction, are impossible using these methods because some semantic information is lost during conversion to formal languages.

Conceptual languages are used in practice because they are well standardized, easier to use than formal languages and have a tremendous amount of support in software systems. Support for the execution of conceptual languages is allowed by the execution languages. Execution languages allow the specification of non-process requirements, such as data structures, forms, and interoperability problems [160]. The BPMN can be converted into Business Process Execution Language (BPEL) [114]. Most business process management systems support the direct execution of BPMN/BPEL models, although the execution behavior may differ slightly from vendor to vendor. Extensive training and best practices on BPMN have been proposed, such as by Bruce Silver [135] and Bernd Ruecker [132].

Low-code platforms use a proprietary conceptual language, typically based on BPMN or flowcharts. For example, the OutSystems platform uses a proprietary BPMN-based syntax called Business Process Technology (BPT) [73]. The absence of a standardized modeling language prevents modelers from leveraging formal analysis when creating software requirements.

To cover business processes that are not flow-oriented but contain ad-hoc activities related to a customer's case, case management conceptual languages are used and supported by case management systems. The conceptual language CMMN [116] was introduced to capture case-related business needs. Case management systems are typical in healthcare, legal, police detective, social work, etc. [105].

## 2.3 Enterprise Engineering

EE [31] aims to assist professionals in mastering the complexity of modern enterprises by providing suitable theories and methodologies. The discipline introduces design and engineering methodology for organizations (DEMO) [32] based on its formal theories. This methodology introduces a manner for modeling that consists of the DEMO specification language and four visual aspect models: the cooperation, action, process, and fact models.

Here, we refer to both the methodology and specification language as "DEMO" and to the four visual aspect models as the "DEMO models".

DEMO models belong to the category of conceptual models from Section 2.2.2. However, because they are based on PSI and DELTA theories [32], they exhibit the qualities of formal languages from Section 2.2.2. Another formal grounding of DEMO models is based on process algebra [148]. Moreover, the DEMO methodology provides a method for creating DEMO models called organization essence revealing (OER) [32]. This method can be applied when collecting process-based compliance system requirements based on human interviews, textual descriptions, and conceptual language descriptions.

Although some studies have applied EE theories and DEMO in software development practice, DEMO itself has been formulated as implementation independent. A notable work in which EE theories are used to tackle rapid IT changes was conducted by Dvořák in [39]. However, despite these efforts, the application of DEMO in software is rare compared with the conceptual languages presented in Section 2.2.2.

## 2.3.1 Ontology

We understand the notion of ontology as a "formal, explicit specification of a conceptualization shared between stakeholders" [59]. The most important criteria regarding the quality of any ontology [61] are i) ontological truthfulness, an ontology providing a truthful representation of the real world; ii) ontological completeness, completeness of expression for any phenomenon that may exist in our domain of the real world; and iii) ontological appropriateness, good support for shared reasoning between stakeholders. Other essential concepts regarding ontologies, conceptual languages, and models are formulated by Guizzardi in [60].

In our work, we use ontology to create a high-quality model of compliance requirements that helps in the transition to the supporting software systems.

## 2.3.2 Important Concepts from PSI Theory

### 2.3.2.1 Actor Roles and Actors

Actors are people who perform a particular role in a business process, such as *aspirant citizens* and *court decision completers*. The DEMO actor role names may seem strange because, in domain-specific texts, no roles such as an *arbitrator complaint meter* or *challenge arbitrator meter* exist. *Plaintiffs* and *judges* are likelier names for these roles. In DEMO, actor roles are abstracted to enable flexible mappings of organizational roles.

### 2.3.2.2 Acts and Facts

Two main types of acts and facts are considered: *coordination* and *production*. Production refers to a product being created. For example, a "court case decision for Marek." Coordination refers to the communication required to reach a shared understanding of the product to be created. An *act* is an action being performed, and a *fact* is the result of

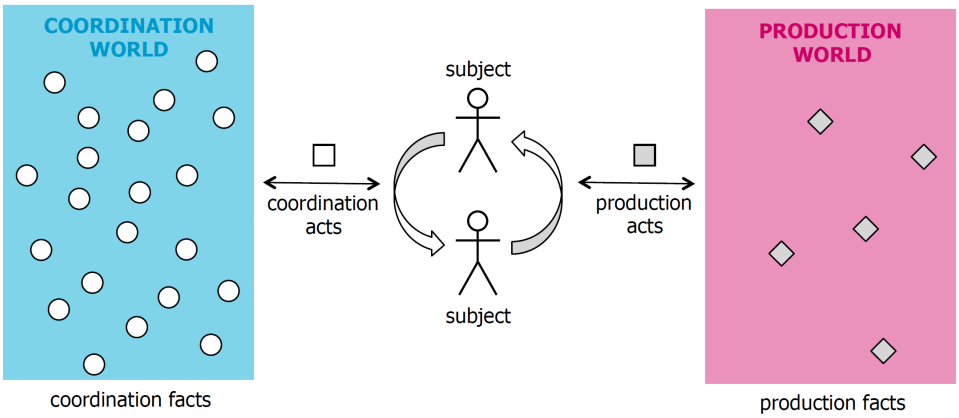


Figure 2.3: Coordination and production worlds [32]

that action. fig. 2.3 shows how acts and facts are created; actors perform acts, and this produces facts.

#### 2.3.2.3 Transaction

The notion of a transaction is the central concept in the DEMO methodology. This is a communication pattern based on Habermas's theory of communicative acts [63]. DEMO uses this pattern to discover communication between people, which guides the modeler to consider all possible outcomes of a communication, not only cases in which everything proceeds in a prototypical manner. The transaction steps are called *coordination acts*, and their completion results are *coordination facts*, as described in Figure 2.3. Figure 2.4 shows the Complete Transaction Pattern, where two abstract *actor roles* progress towards the acceptance (completion) of a transaction. Once a transaction is completed, a *production fact* is produced. A simplified version of the Complete Transaction Pattern is called the Standard Transaction Pattern, which does excludes the four revoke components. For the sake of completeness, an even simpler transaction pattern exists that is called the Basic Transaction Pattern; however, as it contains only the "happy path" request-promise-execute-declare-accept steps, it is not particularly applicable in practice.

DEMO transactions are divided into three categories using the OER method: *ontological*, *infological*, and *datalogical*. TOntological transactions describe original production acts. Infological transactions describe production acts regarding information manipulation to produce ontological products (e.g., computing, inferring, interpreting, reporting, etc.). Datalogical transactions support the infological production acts by storing and retrieving data used by them. The categorization is defined using the distinction axiom and allows the modeler to focus only on the essence of the organization, which consists of ontological transactions. A remarkable property of ontological transactions is that they tend to undergo significantly fewer changes over time than the other two.

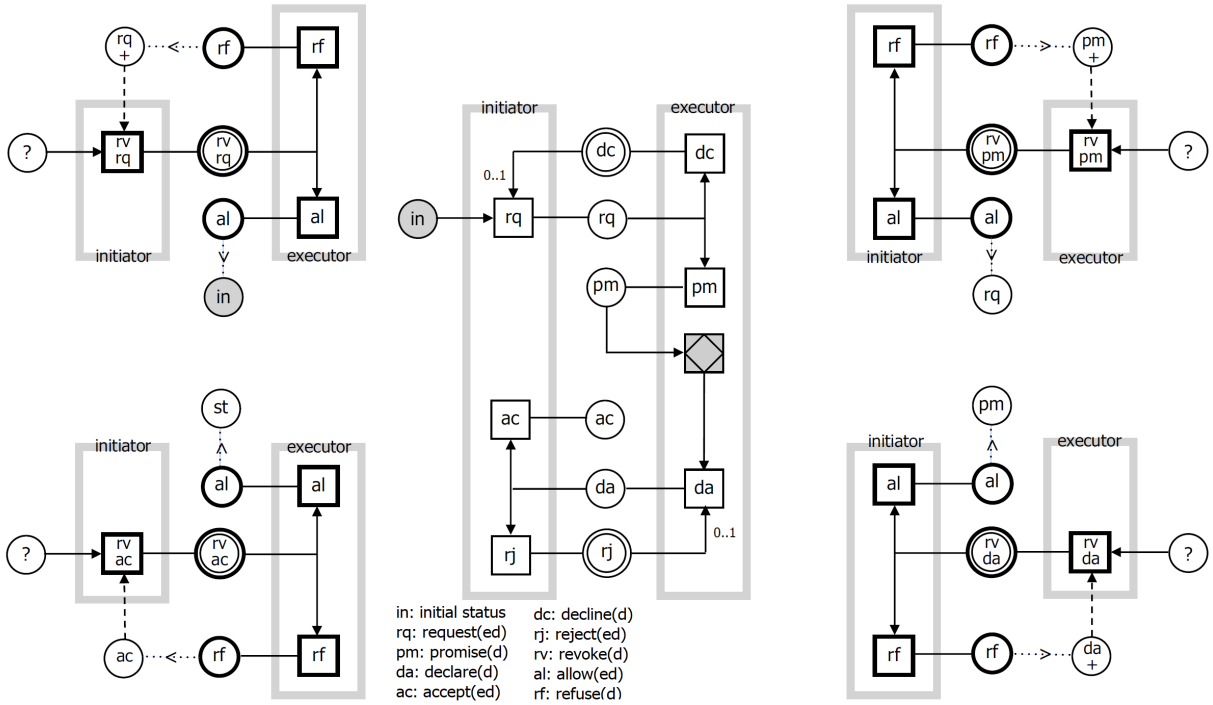


Figure 2.4: Complete Transaction Pattern [32]

#### 2.3.2.4 Transaction Distinction Axiom

The organization layers allow us to distinguish between "important" and "unimportant" transactions. In the context of DEMO, organizational layers allow for the discovery of the essence of an organization.

In Figure 2.5, this distinction of the three categories is applied to a domain description. Original transactions are at the top in red and called ontological. Supporting transactions, such as remembering and sharing, are in green and called infological. Finally, saving and providing information are shown in blue and called datalogical.

Next, we briefly describe how the DEMO methodology can be applied to improve the quality of process-based software requirements. We demonstrate this in the domain of our case studies, procedural law; however, it is applicable to any type of domain aligned with S0.

### 2.3.3 Applying DEMO Methodology to Process Domain Descriptions

The most common manner of describing the processes in an enterprise in practice is to use plain text in a natural language [121]. Based on this observation, the DEMO methodology provides the OER method that guides a modeler in discovering ontological concepts in plain text or a more structured description in the form of conceptual language diagrams, such as a BPMN. The following steps are specified by OER to create a DEMO model.

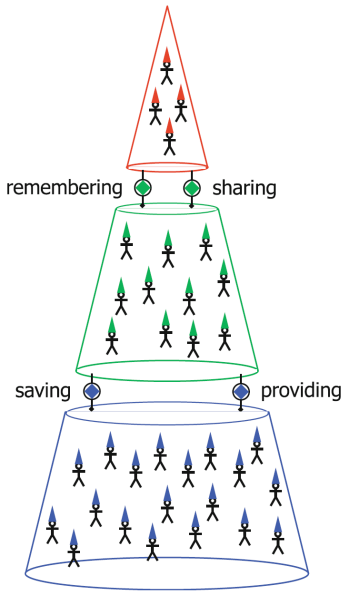


Figure 2.5: Organization layers [32]

1. Revealing the organizational essence using the OER method. This reveals the organization's acts, facts, transaction types, and their distinction according to Section 2.3.2.4.

2. Modeling organizational essence. This is the creation of the four DEMO aspect models.

The following subsections provide a basic introduction to DEMO modeling techniques. Extensive guidance on creating DEMO models can be found in [33], and [32] provides a more complete theoretical grounding.

#### 2.3.3.1 Organization Essence Revealing Method (OER)

The OER method guides the modeler in revealing essential actor roles, transactions, and the structure in which the transactions are organized. The OER method consists of three steps:

1. Identifying the acts and facts,

2. Identifying the transaction types in which these acts/facts occur, and

3. Identifying the decomposition tree-like structures in which these types occur

The OER method can be applied by conducting interviews with domain experts. However, this is also feasible using a textual representation of procedural law that is widely

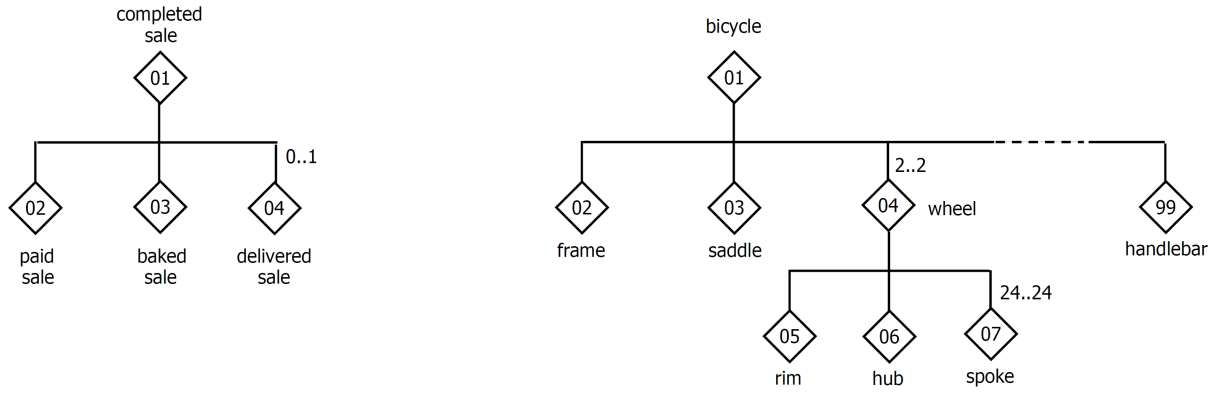


Figure 2.6: Examples of process structures [32]

available. This study used the latter approach, which allows the quality of information contained in an original source to be evaluated.

In the first OER step, a modeler goes through the text and identifies transaction acts represented in the form of verbs and facts. These facts are typically easy to recognize, as the same process would be used to create a data model of the domain. The acts are categorized according to the transaction steps.

The second OER step goes through all identified acts and identifies the transaction types to which the belong. This step can be complicated, as one transaction can begin on page 20, for example, continue on page 2, and end on page 15.

The final OER step identifies the tree composition of the transactions. This determines whether they belong to the same process or if multiple processes are present. This tree composition is a vital aspect of the DEMO methodology compared with the other approaches mentioned in Section 2.2.2. Other approaches conceptualize the process as a flow of activities. However, DEMO conceptualizes it as a tree corresponding to the product structure, which results in better complexity management owing to its hierarchical representation. Figure 2.6 shows an example of such a tree structure.

#### 2.3.3.2 Modeling Organizational Essence

DEMO comprises four different aspect models: the construction model, process model, fact model, and action model. Each represents a view of the same model with different perspectives.

The construction model is the most concise. It shows transaction types, actor roles, the border of the scope of interest, and the transaction-type tree composition. Figure 2.7 shows an example of this model. It presents an excerpt from the arbitration court proceedings discussed later in this thesis (Section 6.2.0.2). The process begins with a claimant initiating the arbitral proceedings transaction. The other transactions in the figure are child transactions according to the composition axiom in Figure 2.6. Compared with a traditional flowchart-based approach that models a process as a sequence of steps, the DEMO model considers the process as a composition of transactions that occur independently

according to the transaction axiom shown in Figure 2.4. For example, after the arbitral proceedings begin, the payment of arbitration fees and acceptance of the settlement can occur simultaneously. This allows for a more realistic representation of the model domain, in which actions typically occur in parallel. In addition, the complete transaction pattern allows for situations in which the process must return because of human errors or people changing their minds. Capturing a sequential (causal) relationship between transactions can be expressed using the PM.

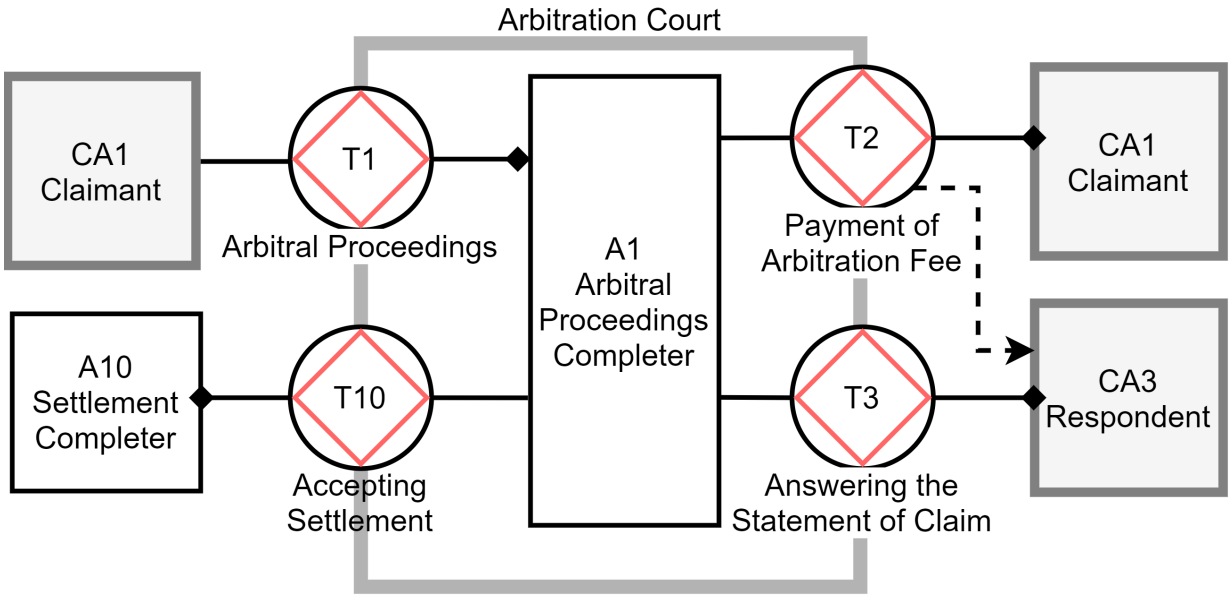


Figure 2.7: Excerpt from an arbitration court process in DEMO the construction model

The process model further specifies the composition of a transaction-type tree. PM involves two concepts. The first specifies the act by which children's transactions are initiated (interaction). The second specifies restrictions between transaction types; for example, a transaction type is not executed before another type (interstriction). This restriction can be applied to transaction types belonging to different processes.

This study expressed the information contained in this model as part of the CM. In Figure 2.7, the conditional link (dashed line) between T2 and CA3 indicates that transaction T3 can begin only after T2 completes. This means that a respondent can answer a claim statement only after the fee for the proceedings is paid.

The fact model is the most intuitive, as it is similar to database models and was adapted from a fact-based modeling used for modelling databases [67, 66]. DEMO simplifies and extends it with reference to process outcomes (products of transactions). It describes the structure of the facts involved in the process, such as an order or customer, their properties, such as age, and the relationship between them. Compared with a typical structural model, it provides a link to a transaction-type result, and therefore links the structure and process behavior. In DEMO methodology, all objects, properties, and relations are referred to as

production facts. Detailing this model further is beyond the scope of this because the study is concerned only with the process.

The action model can be ideologically related to the Object Constraint Language (OCL) in UML [23], which specifies complex business rules such as: "The person needs to be older than 18 years in order to request a beer." The business rules mostly relate to performing coordination acts, that is, decision rules for selecting an appropriate one. The second purpose of this model is to link the FM to the transaction-type steps. Therefore, it generally acts as a "glue" for the other aspect models. Examining the details of this model remains outside the scope of this study.

#### 2.3.3.3 Reduction of Complexity

According to the book [32], EE theories and DEMO methodology reduce complexity by over 95% in terms of the size of model expressions. This is achieved by applying the three concepts mentioned in this section:

1. The transaction pattern as described in Section 2.3.2.3 is used to represent 19 concepts as one concept: transaction. Thus, an average reduction of 80% is achieved [32].

2. The transaction composition axiom demonstrated in Figure 2.6 replaces the flow-based business process models with trees of transactions. This is estimated to reduce the complexity by another 80% [32].

3. The transaction distinction axiom described in Section 2.3.2.4 allows an abstraction from implementation (i.e., user interface, integration, database architecture) and realization (i.e., sharing and storing the information in infological and datalogical layers as described in Section 2.3.2.4). This is estimated to reduce the complexity by another 80% [32]. Guidance towards connecting the implementation and realization to the ontological process model is provided by [28]. Figure 2.8 provides an example of such decomposition, where the top red part resembles the ontological part of the process, and the bottom part models a possible realization of underlying infological and datalogical transactions.

Therefore, we have three dimensions of decomposition that can be used to manage complexity in a similar manner as that in structure analysis/ design [85]. By applying these techniques, the underlying complexity of the domain does not disappear, but is rigorously managed. Therefore, an improvement in the quality of the resulting software specification is expected, which helps in the analysis stage.

### 2.3.4 Ontological Quality

For the evaluation of the process model quality in this study, the C4E quality criteria from EE were used. The definitions are taken from [32].

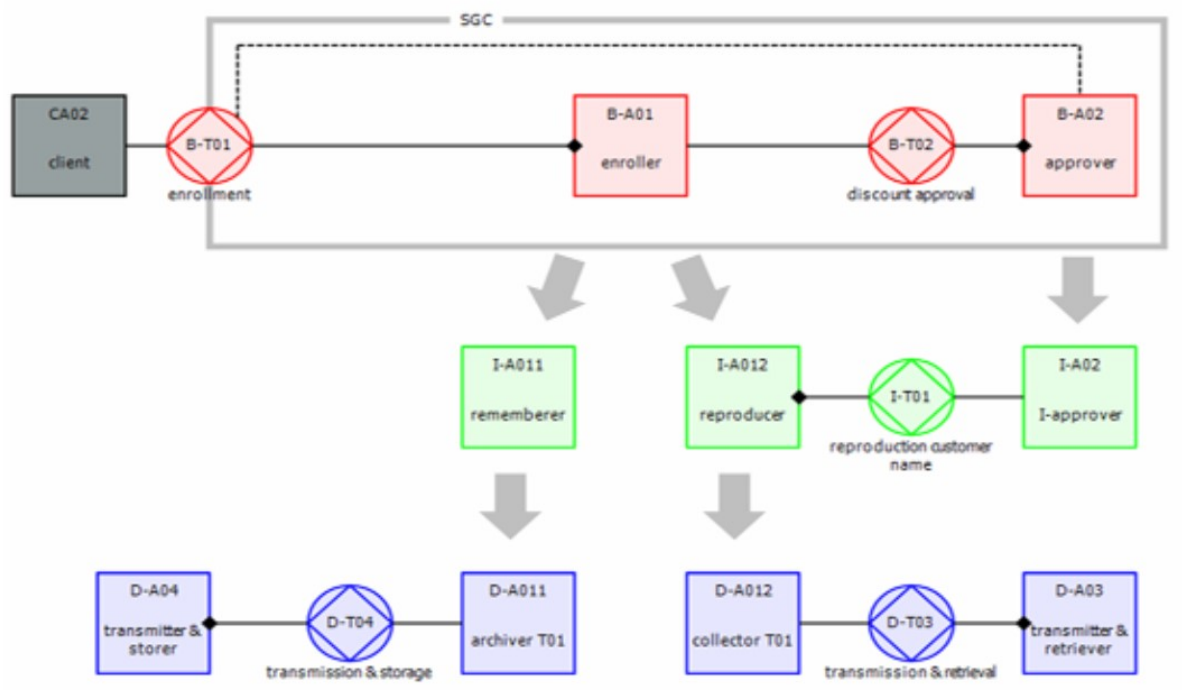


Figure 2.8: Example of DEMO process decomposition according to the transaction distinction axiom [28]

- Comprehensive – means that it is ontologically complete, provided that all knowledge of the concrete system is available. Consequently, this allows for studying the statics of the modeled system (its construction) and its dynamics (its operation).
- Coherent – means that the model elements are connected without 'loose' parts.
- Concise – their size is small compared with current meta models, which mostly do not abstract from realization and implementation.
- Consistent – means that they do not contain logical contradictions.
- Essential – the model is called essential when it satisfies all C4 criteria: comprehensiveness, coherency, conciseness, and consistency. Only one essential model exists for a given domain.

Another quality requirement is that the modeling language should be based on formal definitions. The lack of a precise formal foundation is known as structural ambiguity. In [47], Table 1 provides an overview of formal languages and evaluates their structural ambiguity. DEMO methodology does not suffer from structural ambiguity, because its foundations are defined in the PRISMA formal model [32]. The PRISMA model fulfills the C4E criteria [32], and thus, all valid DEMO models also satisfy the C4E.

### 2.3.5 How to Read DEMO Models

Now that the basic DEMO concepts and models have been introduced, an example execution of the model shown in Figure 2.7 is presented. This provided a better understanding of the models presented in the following section. The simulation aims to show the expressivity of the pattern language, in which each transaction consists of the complete transaction pattern introduced in Figure 2.4. This is a significantly different approach compared with the common flow-chart and BPMN-based diagrams mentioned in Section 2.2.2.

1. Marek is a claimant (CA1) who requests the arbitral proceedings transaction (T1) with his dispute against Company X (product in the DEMO terminology) from Eugene who is an employee of the arbitration Court (thus, given the role of arbitral proceedings completer (A1)).

2. Eugene can act in this process in various manners; he can promise or decline T1 or request T2 and T10. He can perform all actions in parallel. He cannot request T3 because it depends on the completion of T2 (see the dashed line from the process model). According to the complete transaction pattern Figure 2.4, Marek can revoke his dispute if he changes his mind or finds a mistake in his application. The revocation must be accepted by Eugene to detain the process.

3. Eugene chooses to request the payment of arbitration fees (T2) from Marek as a claimant (CA1) of this dispute against Company X. Moreover, he promises Marek that his dispute against Company X will be handled (T1 - promise).

4. Marek promises to pay the fee (T2 - promise) and pays the fee (T2 - declare), and Eugene confirms that the correct amount was received by the arbitration court bank account (T2 - accept).

5. The statement of claim (T3) can now be answered because the arbitration fee was paid (T2 was accepted). Therefore, Eugene requests an answer to the claim statement from Company X (T3 - request) by a certified mail. Company X receives the mail (T3 - promise) and submits a response (T3 - declare). Eugene checks that the response fulfills all formalities and accepts it (T3 - accept).

6. The process continues until the root transaction arbitral proceedings (T1) is accepted. Figure 6.2 and Figure 6.3 present a model of the arbitration court procedure.

## 2.4 Chapter Summary

In this chapter, we introduced the theoretical foundations for our research objective from (Section 1.4). The centralized and decentralized compliance management were elaborated more extensively in the scope relevant to our research. An extensive overview of existing approaches to model and execute the compliance management systems were provided.

Finally, the relevant parts of the enterprise engineering discipline were introduced so they can be related to our research objective and research questions in the following chapters.

# Part III

# Our Approach

Chapter 3

# Research Design

In the Section 1.3 we outlined the problems that we would like to focus on in our research. Later, in Section 1.4 we narrowed the research problems to a single research objective and the research questions. The target domain of our research is highly empirical and therefore we decided to use to use the design science methodology [74, 169]. Furthermore, we adopted the design science methodology to the field of information systems according to [162] and also to the dissertation thesis [161].

In this chapter, we briefly summarize the design science research methodology (Section 3.1), and then introduce a research strategy for our research (Section 3.2). In Section 3.3 the assumptions, scope, and limitations are presented. A role of cooperation with industry in our research is described in Section 3.2.1. Finally, the chapter is summarized in Section 3.4.

## 3.1 Research Methodology

Because of the empirical and multidisciplinary nature of our research objective formulated in Section 1.4, we decided to use the design science research methodology adopted for IS research [74, 169]. To use the methodology correctly, we followed the guidelines for conducting design science research in IS [162] and guidelines for adoption of the design science research to the PhD thesis [161]. This section provides a brief overview of the design science research methodology, in the next section (Section 3.2) the research strategy for our research is presented.

An overview of the design science research framework adopted for IS is shown on Figure 3.1. The most important parts of the framework are the relevance and rigor cycles. The relevance cycle feeds the IS research with business needs and the rigor cycle provides applicable knowledge. Within the IS research, an artefact is built based on the needs an knowledge. After the artefact is build, it is justified and evaluated. In the end, the IS research contributes back to the environment by fulfilling the business needs and back to the knowledge base by additions to the knowledge-base. In our research, the most important

part of the environment are the compliance management systems, and as for the knowledge base, the EE theories.

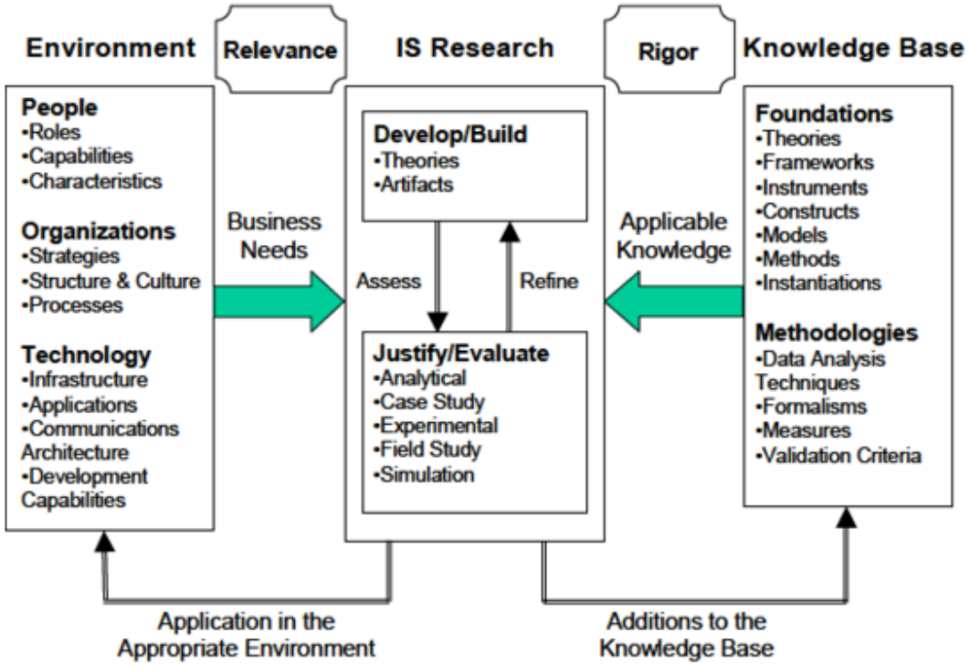


Figure 3.1: Design science research framework adopted for IS research [74, 162]

The research according to the design science is being performed iteratively in DSR cycles. A commonly accepted process to perform such cycle is shown in Figure 3.2. Description of the process steps in the process model was cited from [157]:

1. **Awareness of the problem:** The awareness could be generated from practical experience or from related disciplines. The output from this phase is a proposal.

2. **Suggestion:** The suggestion is closely related to the awareness of the problem (as indicated by the dotted line). The suggestion is often included as a tentative design in the complete proposal as output. However, an approach to develop a suggestion might be included in the proposal if a possible solution is not immediately evident.

3. **Development:** The tentative design is implemented during this phase and the technique for implementation will differ depending on the artefact.

4. **Evaluation:** When the artefact has been developed, the evaluation of the artefact is mandatory, usually according to requirements and criteria specified during the suggestion phase (as part of the proposal). The result of the evaluation should be carefully noted and explained. This phase may result in the refinement of an awareness, a suggestion or a development, especially if the result of the evaluation is not satisfactory.

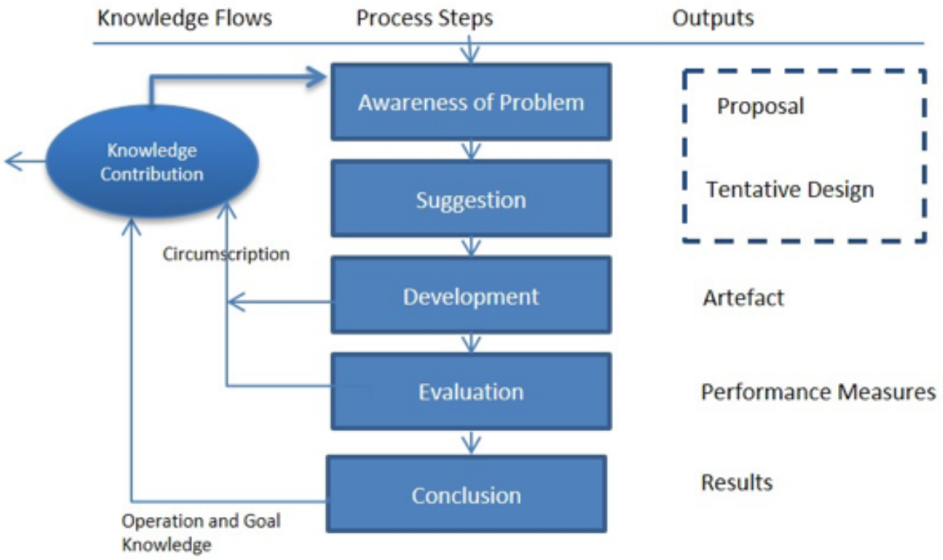


Figure 3.2: Design science research process model (DSR cycle) according to [157]

5. **Conclusion:** This is the final phase when the research results and contribution are identified. This not only includes the artefact, but all additional knowledge with regard to the process, construction and evaluation that were acquired. The output of this phase is an acceptable research contribution.

In the next section, the research strategy for our research is introduced.

## 3.2 Research Strategy

This section presents our research strategy to answer the research objective and research questions defined in Section 1.4. As stated in the previous section, we do applied the DSR for IS research and created a research strategy that contains multiple DSR cycles. This strategy is presented in Figure 3.3 and consists of two main DSR cycles. On the left, there is a cycle for centralized compliance management. On the right, for decentralized compliance management. The business needs for the cycles were presented in the introduction, further narrowed in Section 1.1, Section 1.2, and Section 1.3. The relevance was further strengthened by cooperation with an industry partner as described in Section 3.2.1. Both DSR cycles build on applicable knowledge presented presented in Chapter 2. The research strategy of each DSR cycle is presented in this section. The table Table 3.1 provides a mapping of the research cycles to the research questions formulated in Section 1.4. In the Part IV and Part V, the outputs of the DSR cycles are presented. Finally, in Chapter 9 the findings, evaluation, and contribution of the research is summarized to complete the design science relevance and rigor cycles.

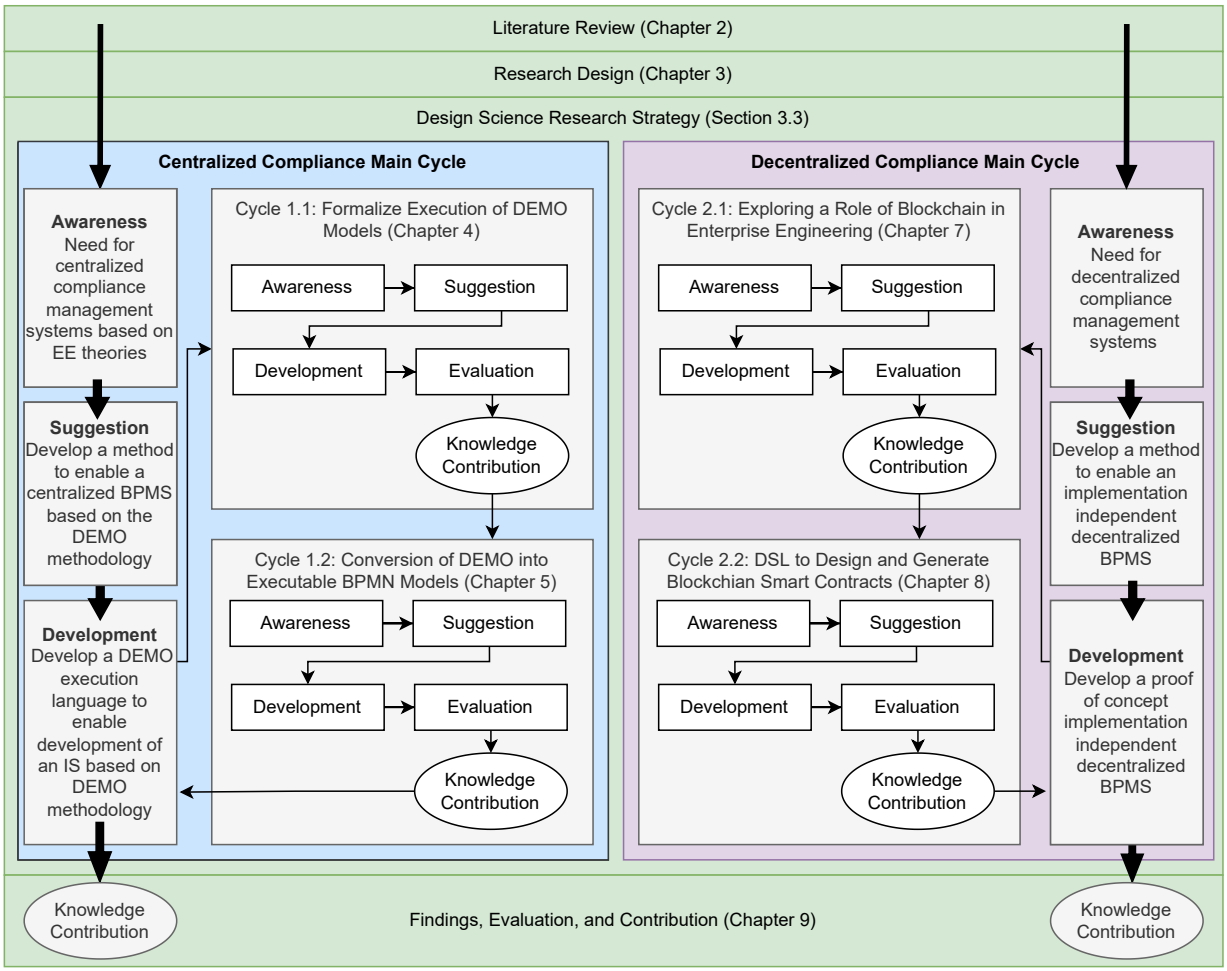


Figure 3.3: Our research strategy

### 3.2.1 Professional Experience

The centralized compliance DSR cycle was done in cooperation with a professional company ForMetis Consultants BV [50]. The company contributed significantly to forming the business needs in the relevance cycle of the research. The main author worked with the company on applying the theories in the appropriate environment. The company's goal was to develop a proprietary BPMS based on DEMO called DEMO engine. The contribution of the main author to the proprietary DEMO engine is outside the scope of this research as they belong to the company. The formalization of execution of the DEMO models called DEMO machine is the main artifact of this research.

### 3.2.2 Centralized Compliance Main Cycle

This research cycle serves to answer research question 1: "How to design software systems to support business process management requirements based on EE theories and DEMO

Table 3.1: Mapping of our research questions and DSR strategy

| Research Question | Centralized Cycle | Cycle 1.1 Chapter 4 | Cycle 1.2 Chapter 5 | Decentralized Cycle | Cycle 2.1 Chapter 7 | Cycle 2.2 Chapter 8 |
|---|---|---|---|---|---|---|
| RQ 1 | X | | | | | |
| SRQ 1 | X | X | | | | |
| SRQ 2 | X | | X | | | |
| RQ 2 | | | | X | | |
| SRQ 3 | | | | | X | |
| SRQ 4 | | | | X | | X |

methodology¿‘ and fulfill the research objective defined in Section 1.4. A DSR process model shown in Figure 3.2 was adopted into a strategy for our research Figure 3.2. Our research strategy for centralized compliance consists of one main DSR cycle and two DSR sub-cycles. The sub-cycles serve to answer the sub-research questions 1 and 2.

The business needs for this cycle were continuously consulted with an industry partner as described in Section 3.2.1. The main objective of this DSR cycle is to find and fill the most important gaps that would enable the development of a BPM system based on DEMO methodology. The most significant gap was identified as a missing execution language for DEMO models. Although a formalization of the DEMO foundations was known in the form of a CRISP model [29], at the time, it was not considered by the industry partners as sufficient for the creation of a DEMO-based BPMS. Therefore, the main artifact created in this cycle is a DEMO execution language we named DEMO machine (Chapter 4). Therefore, this effort has a dedicated sub-research cycle 1.1. Sub-cycle 1.2 explores an alternative path of creating information systems from the DEMO model based on converting the DEMO models into executable BPMN models that can be used in existing BPMS (Chapter 5). Finally, an empirical experiment was proposed to model case studies from the procedural law domain and execute them in software systems(Chapter 6).

### 3.2.3 Decentralized Compliance Main Cycle

This research cycle serves to answer research question 2: "How to digitize business processes using blockchain smart contracts in a methodical way and eliminate programming errors while avoiding a dependency on a particular blockchain implementation¿‘ and fulfill the research objective defined in Section 1.4. The business needs for this research were presented in Section 1.2. The main objective was to explore how to enable decentralized compliance management systems by applying EE theories. Because this topic was not previously explored, a first sub-research cycle 2.1 was created to explore the role of blockchain technology in EE (chapter 7). Based on this sub-cycle, a business need was identified that a DSL to design and generate blockchain smart contracts is needed and was designed as part of the sub-cycle 2.2 (Chapter 8). To evaluate the artifacts created in the sub-cycle 2.2, case studies were made (Section 8.5 and Section 8.6).

## 3.3 Assumptions, Scope, Limitations

This section outlines the assumptions, scope, and limitations of our research.

The main assumption is that the sources we used to outline the challenges of the centralized and decentralized domain in Section 1.1 and Section 1.2 are an accurate representation of the domain. Such claims, although presented by reputable sources, are not possible to verify because the detailed challenges in the domain are related to the proprietary processes of enterprises. Another assumption is that the EE discipline provides theories and methodologies that are superior to the state-of-the-art BPM approaches used in state-of-the-art organizations. Further, we assume that the decentralization of some compliance processes may provide some benefit over the state-of-the-art as suggested by [88].

As for the scope, the target domain of this thesis is very broad, and its challenges are far beyond the scope of a single thesis. Therefore we narrow our focus on finding gaps in the state-of-the approaches and moving them a small step towards their fulfillment. Implementing the software systems suggested in this research is outside of the scope. In some cases, a proof of concept can be made. The same applies to the experiments and empirical case studies. Although our presented case studies contain a considerable improvement in their size compared to similar research, they are still a simplification of the real multi-million $ projects typically made in the target domain.

As for the limitations, this research is focused on the procedural aspect of the compliance processes in compliance management systems. In centralized compliance management, the research is focused on capturing business processes when human cooperation and co-production are taking place. A combination of other software development techniques and deep domain expertise will be required to create production systems. In the case of decentralized compliance, the most significant limitation is the novelty of the field itself. Extensive research highlights the large benefits of using blockchain technology. However, empirical evidence is still missing, and the existing technology platforms are still very immature.

## 3.4 Chapter Summary

This chapter provided a brief overview of our research design. First, the DSR methodology adopted for IS research was introduced. Then a research strategy for our research was presented. Our research strategy consists of two main DSR cycles. The first one focuses on centralized compliance management and enabling of creation of DEMO-based software systems in cooperation with an industry partner. The main results from the cycle are presented in Part IV. The second main cycle focuses on exploring the role of blockchain technology in EE theories and developing a DSL to model blockchain smart contracts. The main results from the second cycle are presented in Part V. Both cycles are evaluated in Chapter 9.

# Part IV

# Centralized Compliance Management

Chapter 4

# Execution of DEMO Aspect Models – FAR Ontology and DEMO Machine

This chapter contains a summary of the preliminary results that were published. The work is focused on bridging the gap between the DEMO methodology and its software implementation. We wanted to understand how the DEMO models are executed and how an IT system can support them. This chapter presents artifacts created during the centralized compliance main DSR cycle and the sub-cycle 1.1. The contents of this chapter were published in [A.1, A.3]. First, the Fact, Agenda, Rule (FAR) Ontology is presented in Section 4.1. Later, an execution language for DEMO models based on the ontology is formalized in Section 4.1. Related research is summarized in Section 4.3. Finally, the chapter is summarized in Section 4.4.

## 4.1 Fact, Agenda, Rule Ontology

FAR Ontology was introduced in [A.1] as an ontological foundation for the DEMO Machine. The FAR Ontology introduces definitions of facts, acts, and rules suitable for software system execution. A brief overview of the most notable contributions is provided in this section.

Facts are defined as factual statements about a world of phenomena. Value of a fact is a valuation function that assigns given fact and transaction instance one of three values - True, False, and Unknown. Facts are devised into three types - internal, external, and composed. The internal facts provide factual statements about the internal state of the DEMO Machine or its history. A grammar is proposed for internal facts to provide a domain-specific language to define such facts. External facts are factual statements about a world outside of the DEMO Machine. External facts are crucial for including any possible domain-specific business logic. Composed facts are facts composed from internal and external facts by using and, or, and not operators. Kleene and Priest's three-valued logic is used for the valuation of composed facts.

Rules do introduce the possibility to perform an automatic transaction state change based on a given fact. This enables the automation of desired steps in a business process.

### 4.1.1 Addressing the DEMOSL-DEMO Machine Deficiencies

Let us elaborate on how the FAR Ontology (and the whole DEMO Machine) may address the challenges stated above.

1. *Integration* and *Facts duplication*. Based on the *Separation of Concerns Principle* from the Normalised Systems Theory [99], the DEMO Machine should not supply the functionality of the already-existing enterprise systems, such as a database. Also, the DEMO Machine should not specify scales, dimensions, sorts, units such as time, money, and others.

2. *Lack of expressiveness*. For areas with already established solutions (like mathematical libraries), these should not be represented in a DEMO Machine to maintain the separation of concerns and the C4-ness criteria.

3. *Modularity* and *Version transparency* are complex topics that cannot be easily commented. They are a subject for future work that should be based on the studies of Normalised Systems Theory mentioned above.

4. *Execution semantics of DEMOSL*. The DEMO Machine should specify the execution semantics. The FAR Ontology focuses on the subset of execution, namely the facts, agenda, and rules concepts.

Let us now dive into the specific part of the DEMO Machine, the FAR Ontology, which will be specified as a set of axiomatic definitions.

### 4.1.2 Fact Axioms

The DEMO theory builds on the Φ theory. The letter Φ stands for "FI", an acronym for Fact and Information about a "world," being a specific part of the universe we are interested in and of which we require factual information or knowledge [29]. Our world of interest is "the world of enterprises". A world of interest is assumed to be composed of Acta, Facta, and Stata. **Stata** are things or phenomena that existed before the beginning of our observation. A **Fact** is a proposition about something that exists in the real world and provides us with *factual knowledge* about the world. Facts can be about either *concrete* or *abstract* things or phenomena. They are the results of **Acta**, being actions or acts undertaken by an entity. Facts come to being by carrying out acts. Once they originate, they cannot disappear; they can only be ignored.

During the design time, we deal with facts as *propositions* about the real world. They exist just as a symbolic structure, and we cannot decide its truthfulness. Then, once the the DEMO Machine executes (i.e. the fact "happens"), we may *valuate* it as `true`, `false`

or `undefined`. Undefined means that the subjects of the proposition do not exist yet, or we do not know the valuation due to, e.g., a technical failure. The valuation may (and typically does) change during the execution. Any calculations based on facts should take this into account. Stata also represents factual knowledge about our world of interest that has existed since the beginning of time. Any facts about Stata are always either true or false.

Let us present the definitions here using the standard mathematical constructs.

**Definition 4.1.1. Fact** A fact is an ordered tuple:

$$Fact := (Identifier, Type, Proposition) \tag{4.1}$$

Identifier – A unique identifier of the fact.
Type $\in \{Internal, External, Composed\}$
Proposition – A specification[1] of the statement about the real world.

**Definition 4.1.2. Value of a fact** is a valuation function:

$$FactValue : (TransactionInstance, Fact) \rightarrow \{True, False, Undefined\} \tag{4.2}$$

**Definition 4.1.3. Transaction Instance Linking (TIL)** is a ternary relation that relates certain transaction instances to each other. This relation is defined outside of the DEMO Machine, which requests this relation to evaluate the rules.

$$\begin{gathered} TransactionInstanceLinking(TIL) := \\ TransactionInstance \times TransactionInstance \times LinkingIdentifier \end{gathered} \tag{4.3}$$

TransactionInstance – A transaction instance unique identifier.
LinkingIdentifier – A name of the relation that holds between the transaction instances.

**Example 4.1.4.** Two transaction instances are sharing the same membership:
("T01_1", "T02_2", "Membership")

**Definition 4.1.5. Internal Fact** is a factual statement about a DEMO model instance.

$$InternalFact : \quad = \quad (Fact, InternalFactExpression) \tag{4.4}$$

**Definition 4.1.6.**

$$\begin{gathered} InternalFactExpression := \\ (singleTransactionComparison)|(multiTransactionComparison) \end{gathered} \tag{4.5}$$

singleTransactionComparison = (transaction).state (operator)
((transaction).state | (state))
multiTransactionComparison = (transactionSelector).(selectorFunction)

[1] FAR does not specify the language, it may be a natural language or any other language.

(t => (singleTransactionComparison))
transaction = this | this.parent
state = *perfect tense intention" as defined in DEMOSL*
operator = == | !=
variable = (transaction).(attribute)
selectorFunction = all | any
transactionSelector = $transactionType < (linkingIdentifier) >$ |
$this.children < transactionType >$
transactionType = *existing transaction type defined in the model*
linkingIdentifier = *identifier of the relation between transactions*

This grammar uses the Extended Backus-Naur Form (EBNF). Round brackets denote non-terminals. Note that the presented grammar is elementary, and it cannot capture all facts about the DEMO model instance or its history. Complete grammar is a subject for further research.

**Example 4.1.7.** Let us show an example by formalizing the fact F02 "Are invoices paid?", which is the situation when all instances of T03 that are linked to the current transaction are in the same state as the current transaction.

F02 = (("F02", "Are invoices paid?"), T03< "$Invoice$" >.all(t => t.state == this.state))

**Definition 4.1.8. External Fact** is a Fact about the world outside the DEMO Machine

$$ExternalFact := (Fact, CalculationEngine) \tag{4.6}$$

CalculationEngine – Identifier of the external system function evaluating the fact.

Data in external data banks are represented as external facts, for instance. External facts represent knowledge of phenomena in the environment that may change over time and have no (known) calculation specification. We operate just with a further unspecified reference to external system function that can evaluate the fact, thus carrying out the separation of concerns principle.

**Example 4.1.9.** A fact that evaluates that the person attached to the transaction instance is older than 18 years

F01 = (("F01", "Is person older than 18 years?"), CalculationEngine)

CalculationEngine may be implemented in any computer technology such as a web service (SOAP or REST) or locally as a system library. In the following code, we implement it as a class in a standard programming language. The calculation of an F01 would be realized as its method:

```
public class CalculationEngine {
[DEMOEngineExternalFact(FactId="F01")]
public FactValue IsPersonEligible (TransactionInstance t) {
var person = DAL.GetPersonByTransactionInstanceId(t.Id);
```

```
if(person == null) return FactValue.Undefined;
else return person.Age > 18 ? FactValue.True : FactValue.False;
}}
```

**Definition 4.1.10. Composed Fact** is a fact composed from internal and external facts.

$$ComposedFact := (Fact, ComposedFactExpression) \tag{4.7}$$

**Definition 4.1.11. Composed Fact Expression**

1. InternalFactIdentifier and ExternalFactIdentifier are composed fact expressions.
2. If $x$ and $y$ are composed fact expressions, then following expressions are also composed fact expressions:
   a) (x and y)
   b) (x or y)
   c) not (x)

Kleene and Priest's three-valued logic is used for the valuation of composed facts.

**Example 4.1.12.** A person is older than 18 years, and he is accepted as an applicant in a membership approval process of the Volley tennis club:

```
F01 = (("F01", "Is person older than 18 years?"),
  VolleyCalculationEngine)
F02 = (("F02", "Is person accepted in the approval process?"),
  VoleyCalculationEngine)
F03 = (("F03", "Is person eligible for membership?"),
  ("F01 and F02"))
```

The resulting truth table is then:

| F01 | F02 | F03 Result |
|---|---|---|
| True | True | True |
| True | False | False |
| True | Undefined | Undefined |
| False | True | False |
| False | False | False |
| False | Undefined | False |
| Undefined | True | Undefined |
| Undefined | False | False |
| Undefined | Undefined | Undefined |

### 4.1.3 Agenda Axioms

An agenda is a set of possible coordination acts (agendum) that is presented to the actor. These are well-defined concepts in the PSI theory. According to the transaction axiom, an actor involved in a transaction is offered to choose one of the valid options to perform coordination acts, which happens in asynchronous time. Example: After a Request from the initiator, the executor may issue either a Promise or a Decline, but other coordination acts such as a Reject are now forbidden to comply with the Transaction Axiom.

An agenda for an actor must be (re)calculated completely at run time by the DEMO Machine of the model instance after each state change of the model instance. It will be shown that causal and conditional dependencies and rules restrict the allowed options for coordination acts. It means that rules are applied to guarantee compliance with the PSI theory. Any extension, enlargement of the transaction transition space, or the state space is impossible since this would violate the PSI theory axioms.

**Definition 4.1.13. Coordination Act (cAct)** is a proposed or intended action for an actor.

$$cAct := (Transaction, TransactionInstance, ActorInstance, Intention, SettlementType) \tag{4.8}$$

Transaction = Transaction kind as defined in DEMOSL.[2]
TransactionInstance = Associated transaction instance. May be empty.
ActorInstance = Associated actor instance.
Intention ∈ {Create(T, n), Promise, Decline, Request, Quit, Accept, Reject, State, Stop, RevokeRequest, AllowRevokeRequest, RefuseRevokeRequest, RevokePromise, AllowRevokePromise, RefuseRevokePromise, RevokeState, AllowRevoke State, RefuseRevokeState, RevokeAccept, AllowRevokeAccept, RefuseRevokeAccept }
SettlementType ∈ { Allow, Enforce, Restrict }

There are two additions to the definition given by the DEMO theory. One is the possibility to create a new transaction (generated by the composition axiom) which the rules will use. $Create(T, n)$ means "Create $n$ transactions of type $T$", where $n$ is a positive whole number. The second is the settlement type which says how the cAct should be dealt with. `Allow` means that an actor is allowed to perform the intention. `Enforce` cAct says that the given intention should be actually performed, unless there is a `Restrict` cAct with the same intention for the same transaction instance. Practically, the `Restrict` cAct also informs the actor why such an intention cannot be performed. In the DEMO theory, an actor can perform an act even when it is restricted. However, in enterprise practice, legal and other compliance are crucial aspects of execution. Thus, we enable this feature in the DEMO Machine.

Please note that in definition 4.1.13 we do not take into account any additional information from inside or outside of the organization. This is due to the separation of concerns

[2]Transaction is also defined by the TransactionInstance if present.

principle addressing the Facts duplication (section 4.1.1). All external information (facts) are handled outside of the DEMO Machine.

**Definition 4.1.14. Agenda** is a function that calculates a set of actor's possible actions based on the current state of the model, taking into account the composition axiom and the respective rules.

$$Agenda : (ModelInstance, ActorInstance) \rightarrow \{cAct\} \tag{4.9}$$

**Definition 4.1.15. Perform cAct**

$$PerformCAct : (ModelInstance, ActorInstance, CActToEnforce) \rightarrow Agenda \tag{4.10}$$

To perform a cAct means that the actor selects an allowed cAct from its agenda and enforces it.

### 4.1.4 Rules and Dependencies Axioms

Rules and dependencies are specifications of either a prescriptive execution of a coordination act or a conditional prohibition of a coordination act for an actor, depending on the evaluation of a fact.

A rule and a dependency restrict the available freedom of an actor to issue coordination acts at the execution time. If the rule or dependency applies, the evaluation takes place at runtime, depending on the state of that model instance. The transaction instance state space and the state transition space of a model instance is further restricted (made smaller). It is impossible to add new options for coordination acts since that would violate the axiomatic specifications derived from the PSI theory.

**Definition 4.1.16. Causal Rule and Dependency** are defined as the application of a rule that results in a transaction state change.

$$CausalRule = (Transaction, TransactionState, Fact, cActTrue, cActFalse) \tag{4.11}$$

**Definition 4.1.17. Evaluation of Causal Rule and Dependency**

```
if TransactionInstance.State == TransactionState
  and FactValue(TransactionInstance, Fact) == True
then anAgenda.Add(cAct(cActTrue, Enforce))
else if False then anAgenda.Add(cAct(cActFalse, Enforce))
```

**Definition 4.1.18. Conditional Rule and Dependency** are defined as the application of a rule that results in a restriction of an agendum, in such a way that one of the allowed coordination acts is prohibited while the rule applies.

$$ConditionalRule = (Transaction, Fact, cActToRestrict) \tag{4.12}$$

Since facts may change over time during execution, a condition that inhibits a specific cAct can be met, and the specific cAct is permitted. If one of two cActs is prohibited in the agenda, then the actor can perform the opposite cAct in asynchronous time. As long as the fact in the conditional rule holds, the actor can't perform the cAct.

**Definition 4.1.19. Evaluation of Conditional Rule and Dependency**

```
if anAgenda(TransactionInstance).Contains(cActToRestrict(Allow))
  and FactValue(TransactionInstance, Fact) != True
then anAgenda.Add(cActToRestrict(Restrict))
```

#### 4.1.4.1 Prohibition or Prescription of an Agenda

The above follows that rules and dependencies operate on an agenda by prohibition or prescription. They reduce the model instance state space and the model instance transition space, which causes a desired limitation of complexity. It is impossible to increase the state and transition spaces by "adding" new options for coordination acts which would be a violation of the PSI theory. Rules and dependencies are calculated immediately during the calculation of the agenda.

### 4.1.5 Discussion and Evaluation of the FAR Ontology

The relation between the FAR Ontology and the DEMO models is as follows. The DEMO models provide a formal specification of the rules and facts created and accepted by stakeholders that represent the enterprise's interaction with its environment. The DEMO Machine specifies the construction of an artifact (a software system) that must fulfill the requirements of the created DEMO models. The FAR Ontology is a crucial part of the DEMO Machine.

The following reasoning is provided to assure:

i A compliance with the PSI theory, the causal and conditional dependencies, and the application of explicitly specified causal and conditional AM rules.

ii A reduction of complexity while maintaining guaranteed ontological conciseness and comprehensiveness.

Assume a model composed of actors and transactions. The application of the Transaction Axiom reduces the number of states of each transaction and the number of states in the model state space, which results in a reduction of complexity.

The application of the Composition Axiom demands that before any production fact can be performed, all child production facts must have been produced, i.e., Stated and Accepted. This further reduces the number of states in the model state space. The ontological conciseness and comprehensiveness of the PSI theory have been shown in [38].

Applying the causal dependencies reduces the state transition space of the model instance since a specific option of an agenda must be chosen while the other agenda options are forbidden.

Conditional dependencies disable specific agenda options until a specific condition has been met. In this way, the state transition space is reduced further, and the state space is also reduced without a loss of ontological conciseness and comprehensiveness.

The DEMO Action Model conditional and causal rules modify the agenda similarly to causal and conditional dependencies, and they reduce the state space and the state transition space further, without any loss of ontological conciseness and comprehensiveness.

For a DEMO Machine based solely but precisely on the PSI theory, it has been argued and shown that there is minimized expression, or zero entropy in expression quality [164]. One and only one model can represent any enterprise that may exist in the real world. In addition, anything that is not an enterprise cannot be represented. Based on this reasoning, it is argued that such a DEMO Machine based on proper implementation of the FAR Ontology will keep these qualities.

## 4.2 DEMO Machine

A DEMO Machine is a theoretical computation engine to simulate DEMO Models. The concept was formally introduced in my master thesis at the Czech Technical University. A proof of concept implementation was also created. In this section, we do summarize the most notable contributions.

Transaction axiom was expressed as a state machine (see Figure 4.1) to precisely define possible state space for a transaction. Composition axiom was defined to define a space state for a composition of transaction kinds. Rule axiom was defined to introduce a deterministic approach to handle state transitions and restrictions in compliance with the other axioms. A high-level overview of agenda calculation based on the axioms is expressed in Algorithm 4.1.

**Definition 4.2.1.** A **DEMO Machine** is an ordered tuple:

$$\begin{aligned} DEMOMachine := (&DEMOEnterpriseApplication, \\ &ExternalFactImplementations, TransactionInstanceLinking, \\ &InputInstructions, OutputMessages) \end{aligned} \tag{4.13}$$

DEMOEnterpriseApplication – A DEMO enterprise application.
TransactionInstanceLinking – Ternary relation that represents connections between transaction instances in the outside world.
ExternalFactImplementations – Outside world implementations of functions that calculate external facts.
InputInstructions – A set of instructions that the machine needs to process.
OutputMessages – Results produced by the machine that represent facts about a behaviour of an enterprise.

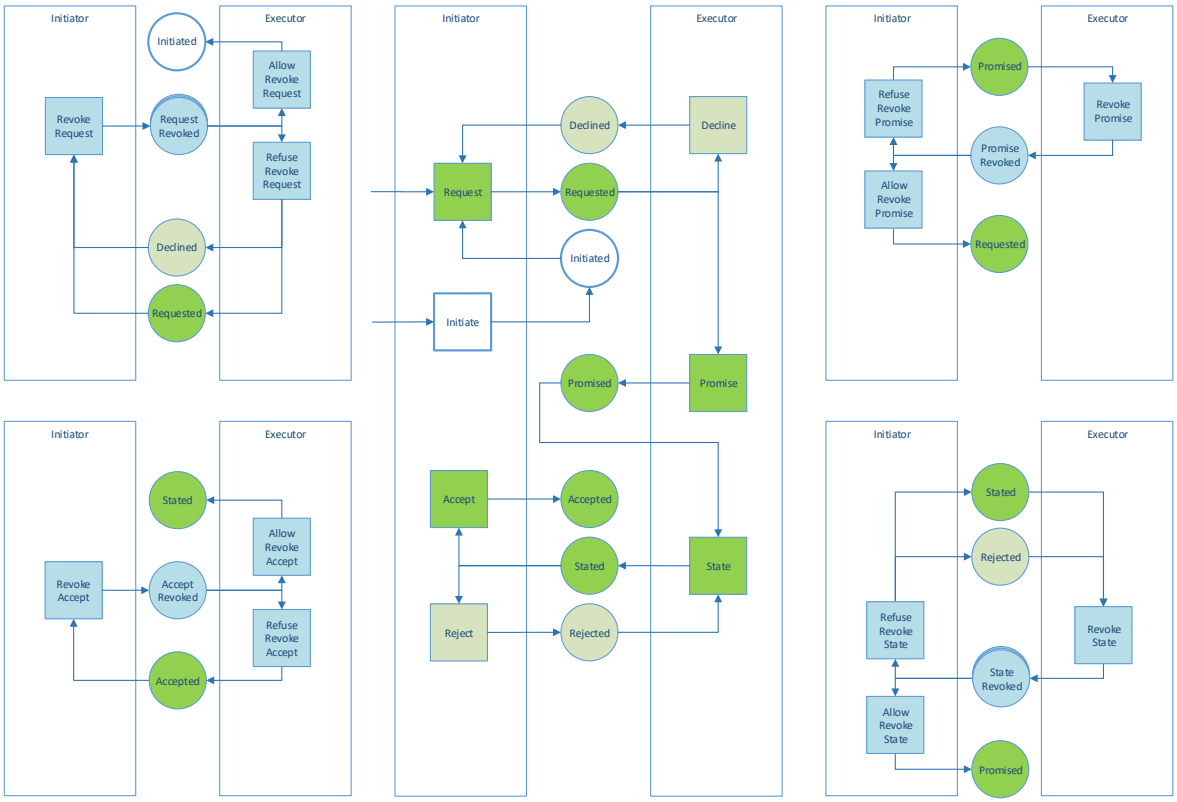


Figure 4.1: Transaction axiom state machine

The DEMO Machine receives instructions on the input and produces messages on the output.

The list of allowed instructions is:

- **GetActorAgenda(Actor)** – Writes an *Agenda* for a specified *Actor* into *OutputMessages*.
- **PerformCAct(cAct)** – Performs a *cAct* and puts a new Agenda for the actor instance (defined in cAct) into *OutputMessages*. Performing an empty cAct causes a recalculation of the model instance.

The Algorithm 4.1 shows a pseudo-code of how the agenda is calculated for a transaction instance.

### 4.2.1 Proof of Concept

In this section, a proof-of-concept DEMO Machine is demonstrated on a Volley club model from the book "The Essence of the Organization" by Jan Dietz [29]. The model is well

**Algorithm 4.1** Agenda calculation

```
1: function CalculateAgenda(transactionInstance, actorPerformCActs)
2:     #Adds actors perform cActs
3:     agenda ←actorPerformCActs
4:     #Adds allowed cActs based on Transaction axiom
5:     agenda.add(TransactionAxiom(agenda))
6:     #Adds allowed and restricted cActs based on Composition axiom
7:     agenda.add(CompositionAxiom(agenda))
8:     #Adds perform and restricted cActs based on Rule axiom
9:     agenda.add(RuleAxiom(agenda))
10:    #Find perform cActs that are allowed and not restricted.
11:    if agenda has cAct c to perform then
12:        #Performs cActs selected to be performed
13:        PerformCAct(transactionInstance, c)
14:        #Transaction states have been changed so recalculation of agenda is needed.
15:        return CalculateAgenda(transactionInstance, nil)
16:    else
17:        #No cActs to be performed found, agenda reached a stable state.
18:        return agenda
```

specified in the book, so we do not elaborate on it much, and we rather point out the differences in our approach and the proposed way of simulation.

We created a proof-of-concept software implementation of the presented DEMO Machine to verify the formal definitions. This section uses a general object-oriented pseudocode inspired by C# to implement the simulation according to the definitions mentioned above.

#### 4.2.1.1 DEMO Model

The organization construction diagram (OCD) in Figure 4.2 contains two transactions describing the situation where a customer comes into the club, requests a membership, pays for it, and he becomes a member.

The Process Diagram (PSD) describes how are the two transactions related. The membership payment is requested after a membership start is promised. There is also a conditional link that specifies that the membership execution phase cannot be done until the membership payment is accepted. Cardinality is not mentioned here, but we expect only one payment per membership. Later payments are not part of the model.

The Action Model (AM) here consists of four rules, all of which are for the membership starter (A1). The logic of working with facts defined in the OFD is also included in the rules, but DEMO materials do not elaborate on how they should be dealt with. The precise definition of executing these AM rules is also not provided, but this notation is sufficient for communication between human stakeholders.

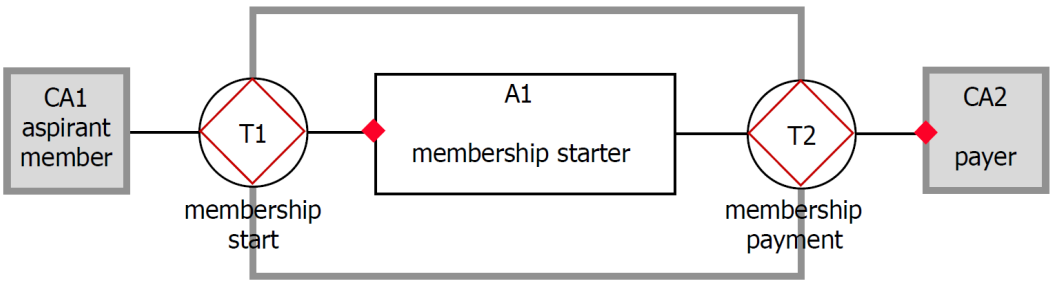


Figure 4.2: OCD model of the Volley Club [29]

1. **Action Rule for A1(1)** – When the membership start (T1) is requested, in case the person who is requesting is eligible, then it is automatically promised, otherwise declined. Eligibility means that the person is old enough, starting day of the membership is the first day of some month, and a maximum number of members was not reached.

2. **Action Rule for A1(2)** – When the membership start (T1) is promised, then automatically request the membership payment.

3. **Action Rule for A1(3)** – When the membership payment (T2) is stated, while the paid amount for the membership has been paid, then accept the membership payment (T2), otherwise reject (T2).

4. **Action Rule for A1(4)** – When the membership start (T1) is promised while the membership payment (T2) is accepted, then execute the membership start (T1) and state the membership start (T2).

#### 4.2.1.2 DEMO Machine Model

Here is what the same Volley club model looks like when described by the concepts introduced in this paper.

OCD and PSD remain the same. They are represented as:

```
AspirantMember = ("Aspirant member", Composite);
MembershipStarter = ("Membership starter", Elementary);
Payer = ("Payer", Composite);
T1 = ("T01", "Membership Start", MembershipStarter, {AspirantMember})
T2 = ("T02", "Membership Payment", Payer, {MembershipStarter})
VolleyClubModel = ("Volley Club", {T1, T2}, {AspirantMember, MembershipStarter,
    Payer}, ...)
```

Information about memberships or persons is likely to be stored in an external database, and there is no use in duplicating them inside the DEMO Machine, as explained in Section 4.1.

The action model implementation differs from the DEMO, so let us go through the Volley club business rules and see how they are expressed in the DEMO Machine.

**Action Rule for A1(1)** is represented by an external fact and a causal rule. The external fact contains all the business conditions needed to evaluate whether a person is eligible for membership. The *LogicalProposition* is there merely to suggest what logic should be used to evaluate such fact. The actual logic then lies in the outside world implementation, and it calls the database. A benefit of this approach is that we do not need to change the model when this business rule is modified. A new implementation version is simply plugged in, and the system goes on.

The causal rule *T1RequestedCausalRule* is there to implement the action part (state transition) of the AM rule. It says: "When an instance of *transaction1* is in state Requested and fact *IsMemberElegibleFact* is evaluated as True, then add a cAct with SettlementType=Perform and Intention=Promise to the transaction instance agenda. If the fact is evaluated as False, then add a cAct with SettlementType=Perform and Intention=Decline to the transaction instance agenda." This explanation may seem more complicated than the previous action rule, but it covers many more scenarios. Adding an enforcing cAct is used instead of a direct state transition because some conditional rule may forbid the transition. The state transition also needs to be allowed by the transaction or the composition axiom. If multiple rules enforce different state transitions, a priority should be assigned to the rules.

```
IsMemberElegibleFact = ExternalFact("Is member eligible for application?",
  LogicalProposition = "Person.Age >= Minimal_Required_Age",
  VolleyClubCalculationEngineId)
T1RequestedCausalRule = CausalRule(T1, Requested, IsMemberElegibleFact,
  cAct(T1, T1.Current, T1.Current.Executor, Promise, Perform),
  cAct(T1, T1.Current, T1.Current.Executor, Decline, Perform))
```

**Action Rule for A1(2)** – is represented by a causal rule and an external fact. The external fact will always be True in this case since there are no business conditions. The causal rule is expressed below, and it says: "If the transaction instance of type T1 is in state Promised and the *TrueExternalFact* is evaluated as True, then add a cAct that (i) performs the creation of a new instance of T2 that will be a child of the current T1 transaction instance and (ii) will be in state Created to the current transaction instance agenda".

```
T1_Promised_CausalRule = (T1, Promised, TrueExternalFact, cAct(T1, T1.Current,
    T1.Current.Executor, Create(T2, 1), Perform), null)
```

Interestingly, the transaction instance T1 can get into state Promised multiple times. Does it mean that it should create a new instance of T2 each time it gets there? Moreover, does it depend on some external system? In this model, the creation of unwanted transactions is controlled by the 1..1 cardinality defined in the PSD. However, for generic purposes, we introduced a possibility for external fact implementation to return several transactions to be created together with the fact result. This is the way how we can control how many transactions are created.

Another problem is determining the executor actor instance for a created transaction instance of T2. It is evident in this particular model that the membership payer will be the same person as an aspirant member. However, it is not formally defined. We do delegate this problem to the outside world implementation. Once it is notified about the created instance of T2, it has all the information it needs to assign the executor. More practical experience shows whether this is sufficient or a more sophisticated solution needs to be designed.

This proof of concept implementation does not contain the composition axiom, and therefore the rules to create child transactions are not implemented as well.

**Action Rule for A1(3)** – is represented by a causal rule and an external fact. The external fact is a business rule determining whether the paid amount was enough. The causal rule then performs accept or reject.

```
IsPaidAmountEnoughFact = ExternalFact("Is paid amount for membership enough?",
  LogicalProposition = "this.Membership.AmountToPay <= this.Membership.Payment.
     AmountPaid", VolleyClubCalculationEngineId)
T2StatedCausalRule = CausalRule(T2, Requested, IsPaidAmountEnoughFact,
  cAct(T1, T2.Current, T2.Current.Executor, Accept, Perform),
  cAct(T2, T2.Current, T2.Current.Executor, Reject, Perform))
```

**Action Rule for A1(4)** – is represented by a conditional rule and a communication fact. We only want the execution phase to be allowed when the child transaction of the T1 instance is in state Allowed. We capture such fact using a communication fact that says: "Are all current transaction instance children with type T2 accepted?". If there is no child transaction with type T2, the fact is evaluated as Undefined.

The conditional rule says: "If there is a cAct with Intention=State and Settlement-Type=Allow within the current transaction instance agenda and the fact *IsMembership-PaidFact* is not evaluated as true, then a cAct with Intention=State and Settlement-Type=Restrict is added to the current transaction agenda." Simply put, the transaction instance state Stated can only be reached when the fact is True.

```
IsMembershipPaidFact = CommunicationFact("Is membership paid?",
  CommunicationFactExpression = "this.children<T02>.all(t => t.state == accepted
     )", VolleyClubCalculationEngineId)
T2StatedCausalRule = ConditionalRule(T1, IsMembershipPaidFact, State)
```

#### 4.2.1.3 Volley Club Outside World Implementation

The outside world consists of the implementation of external facts, transaction relation provider, and state change receiver. It can be implemented in any programming language as long as it provides the values required in the definitions. In our proof of concept implementation, we created a simple implementation of such system that accessed a database and returned relevant values. However, a detailed description of such implementation is not relevant for the purposes of this thesis.

#### 4.2.1.4 Step by Step Execution

In this section, we will provide a detailed description of what happens in the execution of the Volley club model during the happy-flow scenario.

At first, a Volley club enterprise application is created, and implementation of the outside world is attached. The Activity Log shows all the changes in the running enterprise application, and we present all steps of the simulation below.

**Step 1** – We create enterprise positions and attach them to actor roles. Then we assign actors to enterprise positions. Marek is going to be a Customer, which is an enterprise position with actor roles *Aspirant member* and *Payer*. Elisabeth will be an Employee – the Membership starter since she works in the Volley club.

**Step 2** – Marek would like to be a member of the Volley club, so he initiates a new transaction 1 instance and selects its executor to be Elisabeth. He is prepared to request membership, but he needs to fill out the starting day. He fills today and performs the request. Because there is nothing to restrict Marek's request, the transaction moves to the state Requested. New *Membership* object is created, and it stores the data Marek entered. In state T1 Requested, a causal rule is defined and therefore evaluated. Marek is 27 years old, and that is enough to be a member of the Volley club. The causal rule adds enforcing cAct to the agenda, and it moves the transaction to state Promised. In the Promised state, a conditional rule restricts the State from being performed before the membership is paid. The communication fact is evaluated as Undefined because there is no accepted child T2. No interaction was required from Elisabeth.

```
New transaction T01 was created with name=T01.1.
T01.1:Request:Allow
Initiator of T01.1 performed Request.
T01.1:Request:Allow,T01.1:Request:Perform
Fact "Is member eligible for application?" was evaluated as True.
T01.1:Promise:Allow,T01.1:Decline:Allow,T01.1:RevokeRequest:Allow,T01.1:Promise:
    Perform
Fact "Is membership paid?" was evaluated as Undefined.
T01.1:State:Allow,T01.1:RevokePromise:Allow,T01.1:State:Restrict
```

**Step 3** – Elisabeth received a request from Marek, and she would like to deliver him the membership. However, she needs to ask for a payment first, and therefore she initiates a new transaction 2. After transaction 2 was initiated, the conditional rule was evaluated again. Now, the result of communication is not Undefined but False. This is because the T2 exists.

```
New transaction T02 was created with name=T02.2.
Fact "Is membership paid?" was evaluated as False.
T01.1:State:Allow,T01.1:RevokePromise:Allow,T02.2:Request:Allow,T01.1:State:
    Restrict
```

**Step 4** – Elisabeth calculated a membership fee for Marek, and she requested a membership payment. The communication fact is still False.

```
Initiator of T02.2 performed Request.
Fact "Is membership paid?" was evaluated as False.
T01.1:State:Allow,T01.1:RevokePromise:Allow,T02.2:Request:Allow,T01.1:State:
    Restrict,T02.2:Request:Perform
Fact "Is membership paid?" was evaluated as False.
T01.1:State:Allow,T01.1:RevokePromise:Allow,T02.2:Promise:Allow,T02.2:Decline:
    Allow,T02.2:RevokeRequest:Allow,T01.1:State:Restrict
```

**Step 5** – Marek promises to pay for the membership. Before he states the payment, he needs to fill the amount to pay based on the requested amount created by Elisabeth. He fills in 30 euros and states the payment. When transaction 2 is stated, a causal rule that validates if the paid amount is valid is activated. The sum of money matches, and transaction 2 is accepted. Communication fact "Is membership paid?" is finally evaluated as True.

```
The executor of T02.2 performed Promise.
Fact "Is membership paid?" was evaluated as False.
T01.1:State:Allow,T01.1:RevokePromise:Allow,T02.2:Promise:Allow,T02.2:Decline:
    Allow,T02.2:RevokeRequest:Allow,T01.1:State:Restrict,T02.2:Promise:Perform
Fact "Is membership paid?" was evaluated as False.
T01.1:State:Allow,T01.1:RevokePromise:Allow,T02.2:State:Allow,T02.2:
    RevokePromise:Allow,T01.1:State:Restrict
The executor of T02.2 performed State.
Fact "Is membership paid?" was evaluated as False.
T01.1:State:Allow,T01.1:RevokePromise:Allow,T02.2:State:Allow,T02.2:
    RevokePromise:Allow,T01.1:State:Restrict,T02.2:State:Perform
Fact "Is paid amount for membership enough?" was evaluated as True.
Fact "Is membership paid?" was evaluated as False.
T01.1:State:Allow,T01.1:RevokePromise:Allow,T02.2:Accept:Allow,T02.2:Reject:
    Allow,T02.2:RevokeState:Allow,T02.2:Accept:Perform,T01.1:State:Restrict
Fact "Is membership paid?" was evaluated as True.
T01.1:State:Allow,T01.1:RevokePromise:Allow,T02.2:RevokeAccept:Allow
```

**Step 6** – Elisabeth is allowed to state the membership, and she does so. The communication fact "Is membership paid?" was evaluated once more because transaction 2 could have changed in the meantime.

```
Executor of T01.1 performed State.
Fact "Is membership paid?" was evaluated as True.
T01.1:State:Allow,T01.1:RevokePromise:Allow,T02.2:RevokeAccept:Allow,T01.1:State
    :Perform
T01.1:Accept:Allow,T01.1:Reject:Allow,T01.1:RevokeState:Allow,T02.2:RevokeAccept
    :Allow
```

**Step 7** – Marek accepts the membership creation.

```
Initiator of T01.1 performed Accept.
T01.1:Accept:Allow,T01.1:Reject:Allow,T01.1:RevokeState:Allow,T02.2:RevokeAccept
    :Allow,T01.1:Accept:Perform
```

```
T01.1:RevokeAccept:Allow,T02.2:RevokeAccept:Allow
```

**Step 8** - Marek is a proud member of the Volley club. We can see that his record was created in the database. The *TransactionId* is there to associate the DEMO engine transaction instance identifier with the membership record. The relation could also be stored inside the DEMO engine as the transaction instance's external identifier.

## 4.3 Related Research

### 4.3.1 The DEMO Engine and the Enterprise Operating System

DEMO Engine of the ForMetis Consultants company is a software system for designing DEMO models with the ability to simulate DEMO models for validation and to provide model execution in full production [86]. Construction of DEMO models is done using the graphical representation of the DEMO OCD in a graphical environment. In the current implementation, the DEMO Process Model is primarily calculated from the OCD. Response links and waiting links (causal and conditional dependencies) can be then specified using the graphical representation of the PSD. There is a limited and not well-engineered support for even simple Action Model rules, which is the aim of our FAR Ontology (Section 4.1).

The Enterprise Operating System [164] is software system composed of a set of DEMO models and a DEMO model executing software engine, the DEMO Engine. The EOS captures and controls all phenomena that occur in operation of the organizational business transactions. This is very similar to an operating system of a computer that reads from and writes to binary registers of a CPU and peripheral controllers and supports many tasks. Using a computer without an operating system is extremely difficult and error-prone. This seems to apply also to controlling and monitoring enterprises without an appropriate enterprise operating system.

### 4.3.2 DEMOBAKER

There is an approach to formalize the simulation of DEMO models and support their execution with a software tools called DEMOBAKER – A New Action Rule Syntax for DEmo MOdels Based Automatic worKflow procEss geneRation [49]. Their approach is based on extending the action model syntax so the DEMO models are suitable for easy transformation to BPMN and BPMS. We do support this approach because existing state of the art BPMS can benefit from DEMO methodology. However to unleash a full potential of enterprise engineering a DEMO-based BPMS based on proper foundations needs to be introduced.

### 4.3.3 XModel

The solution of devising a workflow software system based on model presented by Johanndeiter et al. in [84] is based on the OrgML modelling language, a part of the MEMO

framework, and the XMF metaprogramming platform. The idea is also based on applying the MDE approach, while avoiding the error-prone manual coding stage. The idea is based on applying multiple levels of meta-modelling and utilising XMF's unique features to support multiple dynamic levels of abstraction. The approach seems very interesting, however it seems to lack a proper evaluation in enterprise. Our approach also differs in a careful selection of ontologically well-founded methodologies that exhibit necessary qualities and benefits.

## 4.4 Chapter Summary

In this chapter, we proposed a theoretical computation model called the DEMO Machine, and we demonstrated its capability to simulate DEMO models on a Volley club example. The evaluation of the research and its application in practice is discussed in Chapter 9.

Chapter 5

# Converting DEMO PSI Transaction Pattern into BPMN: A Complete Method

The DEMO methodology is most often used during business processes' strategic and analytical modeling. There are usually two options when it comes to implementing the process using an IT system. The first one is that a business process model is a part of the requirements specifications for an IT system. In the second one, the business process model serves as a source code for the supporting IT system. This approach is well-established in existing BPM systems. However, most existing systems use the BPMN (Business Process Model and Notation) [114] graphical notation standardized by OMG. Therefore, we present an original method for converting enterprise ontology Design & Engineering Method for Organisations (DEMO) process models into a BPMN 2.0 notation. By this approach, we can mitigate certain methodological deficiencies of BPMN. The method exhibits the following qualities: Implementation of the complete transaction pattern formulated by the PSI theory, correct management of multiple child transaction instances, and executability of the resulting BPMN model.

The content of this chapter was published at EEWC conference [A.2]. The results are related to the DSR cycle 1.2 and sub-research question 2. The understanding of executing DEMO models is based on the DEMO Machine introduced in Chapter 4. The method was later automated in the work of Štěpán Tužil [155].

## 5.1 Introduction

BPMN (Business Process Model and Notation) [114] is a graphical notation that is used for modeling business processes. Critical characteristics of BPMN are simplicity of the underlying theory (flowchart), standardized notation, and a large number of tools. This makes BPMN one of the most widespread process modeling notations in practice, despite

its limitations and flaws. BPMN offers three different types of diagrams: Choreography, Conversation, and Collaboration diagrams. For this work, only the Collaboration Diagram will be considered. This diagram expresses the process flow in achieving participants' goals.

One of the BPMN weaknesses is the absence of a methodology for constructing diagrams, which is addressed, for example, by Silver [135]. Nevertheless, the design freedom is still too broad, which results in different modeling styles of individual analysts and different models depicting the same situation, which complicates enterprise engineering tasks like mergers and reorganizations.

DEMO (Design & Engineering Method for Organisations) [29] is a leading modeling method used in the discipline of Enterprise Engineering [31] based on a deep and sound theoretical basis (the PSI-theory) and high ontological relevance. Its benefits for the practical use has been proven, as documented for example in [118] or [40]. It does not limit itself to process modeling, but it also deals with capturing structural (factual) knowledge and business rules, thus delivering a complete enterprise ontology exhibiting certain criteria (C4E). However, DEMO is still a niche approach and relatively demanding to master compared to BPMN. Also, a limited number of tools are available today.

For a brief description of DEMO, we take a help of Op't Land and Dietz [118]:

A complete, so-called essential model of an organization consists of four aspect models: *Construction Model (CM)*, *Process Model (PM)*, *Action Model (AM)*, and *Fact Model (FM)*. The CM specifies the composition, the environment, and the structure of the organization. It contains the identified *transaction types*, the associated *actor roles* as well as the information links between actor roles and transaction banks (the conceptual containers of the process history). The PM details each transaction type according to the *universal transaction pattern*. In addition, it shows the structure of the identified business processes, which are trees of transactions. The AM specifies the imperatively formulated *business rules* that serve as guidelines for the actors in dealing with their agenda. The FM specifies the *object classes*, the *fact types* and the declarative formulations of the *business rules*.

The DEMO Process Model reveals details of the transactions with the respect to universal transaction pattern. The basis is the "happy flow" consisting of `request`, `promise`, `state` and `accept`, which is also called the *basic transaction pattern*. In the so-called *standard transaction pattern* (not depicted), `decline` may happen instead of `promise` and `reject` may happen instead of accept. Then, a new attempt may be made, or `quit`, resp. `stop` may end the transaction unsuccessfully. Real situations may become even more complicated, which is addressed by the *complete transaction pattern* in fig. 5.1. It incorporates the notion of *revocation* – an actor may want to "take back" their act done before[1]. If that is allowed by the other party, the transaction "rolls back" to the desired state.

The logic of the complete transaction pattern is automatically included in all DEMO transactions, which is one of the reasons why the models are compact.

The main goal of this paper is to combine the simplicity of the BPMN and the ontological qualities of the DEMO. The result is the method that converts enterprise ontology

[1]In the DEMO theory, nothing can disappear, so the original fact remains in the fact bank. However, the transaction flow is changed.

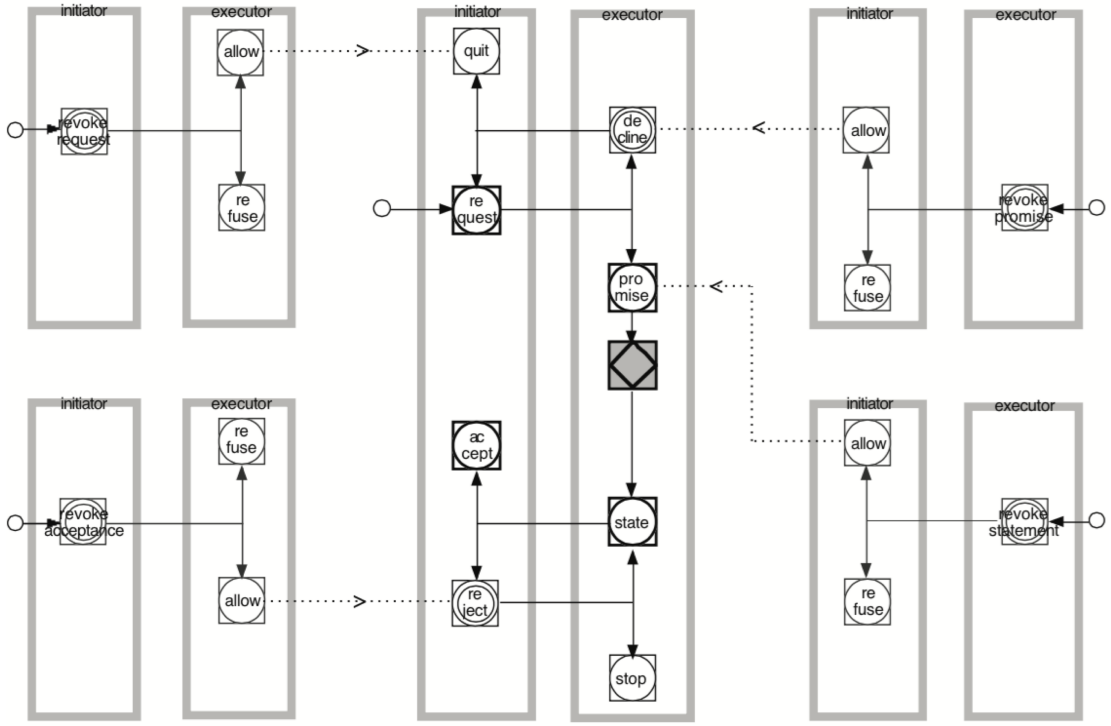


Figure 5.1: DEMO complete transaction pattern [33]

Design & Engineering Method for Organisations (DEMO) process models into a BPMN 2.0 notation. This approach mitigates the mentioned absence of a sound methodological approach for BPMN. The BPMN models resulting from the described method converge, similarly to DEMO, to one essential model, thus eliminating different modeling styles of individual analysts leading to comparable models. Our other requirements are: implementation of the complete transaction pattern formulated by the PSI theory, correct managing of multiple child transaction instances, and executability of the resulting BPMN model.

We start the paper by the discussion of the related work of efforts of improving BPM and BPMN, specifically, the approaches based on applying the enterprise-engineering rigor (section 5.2). We then briefly present the results of a comparative analysis of DEMO and BPMN (section 5.3), which led to formulating our method of conversion (section 5.4). We demonstrate the method on an example (section 5.5). Finally, we discuss the result and formulate conclusions (section 5.6).

## 5.2 Related Work - Improving BPM and BPMN

Poor ontological quality of BPMN is generally known and documented [62]. The most practiced remedy is exercising a methodological approach like the one proposed by Silver [135], who distinguishes three levels of BPMN: (i) Descriptive, (ii) Analytical, and (iii) Executable and proposes several analysis patterns and anti-patterns.

The discipline of enterprise engineering (EE) [31] brought about a rigorous approach of building an enterprise ontology (EO) [29], DEMO being its modeling method. There are several foundational EE theories, the most notable being the PSI theory. As one of the central concerns of EE is the business process management, the effort to apply EE theories (EET) to the existing (less formal approaches) is promising. The efforts in this area are twofold:

1. Applying EET for analysis of existing BPMN models of business processes: for example [16], [165] and [113].
2. Enhancing the formal foundations of BPMN by EET: for example [16], [165], [49], [72]

### 5.2.1 Applying EET for Analysis of Existing BPMN Models of Business Processes

Caetano et al. showed that applying the DEMO PSI-theory to improve business process modeling deserves attention [16]. The authors analyzed existing BPMN models and identified missing DEMO transaction pattern steps in these models. It had been determined, or each BPMN activity from the analyzed models, whether this activity is an ontological, infological, or datalogical part of a transaction. It was also determined which part of the transaction pattern each activity represents. Next, the authors created an ATD and a PSD diagram of DEMO, and using a PSD diagram, they enriched existing BPMN models by adding missing parts of the transaction pattern into the BPMN models.

In the second part, the authors present results of applying this method to analysis of existing BPMN models of key processes of a big organization (more than 500 activities and 60 actors). The authors identified numerous missing act types in the original BPMN models. The results from this analysis were: (i) 25% of production C-acts missing in the original BPMN model, (ii) 25% of request C-acts missing in the original BPMN model, (iii) 50% of promise C-acts missing in the original BPMN model, (iv) 25% of state C-acts missing in the original BPMN model, (v) 40% of accept C-acts missing in the original BPMN model.

Results reported by Pergl and Náplava for an academic institution [113] state reduction of DEMO essential models complexity to 21% of the original BPMN size and several model quality improvements similar to [16].

### 5.2.2 Enhancing the Formal Foundations of BPMN by EET

These efforts aim to express the EE ontological constructs precisely using the standard BPMN notation. Two approaches have been followed. The first is to enhance the BPMN models by adding the missing C-(F)acts and other constructs from the PSI-theory. Caetano [16] is an example of this method.

The second way is generating BPMN models from the DEMO models. This method was discussed in the diploma theses [72], from which the approach in this paper was designed.

## 5.3 Analysis of DEMO and BPMN

Here follow observations of comparing various aspects of DEMO with respect to BPMN, from which follows the conversion principles and decisions made. These were formulated based on the DEMO theory axioms and models definitions related to the BPMN elements definitions, as introduced in section 5.1.

- Similar parts of methods that can be simply transformed from the DEMO to BPMN:
  - The *Process Structure Diagram (PSD)* of DEMO contains process information, which can be related to a BPMN process diagram.
  - The *Action Model (AM)* of DEMO expresses complex decision rules for Coordination acts (C-acts)[2]. The contained information can be used for branching in BPMN.
  - BPMN does not distinguish the three key human abilities (forma, informa, performa), however applying this distinction can be introduced straightforwardly, as shown for example in [113]. As this concern is orthogonal to our effort, we do not discuss the distinction axiom here.
  - Related to the point above, the (atomic) actor roles in DEMO are executors of exactly one transaction, while swimlanes may contain many different actions.
- Different parts of methods that require deep analysis before transformation from DEMO to BPMN:
  - The DEMO *Transaction Axiom* concept does not exist in BPMN. Only happy flows and the most apparent unhappy flows are expressed in models.
  - The *Object Fact Diagram (OFD)* being a factual model does not have an analogy in BPMN.
  - DEMO and BPMN employ different execution models. While BPMN is flow-based, DEMO operates based on a so-called CRISP model [29], which may be characterized as an event-driven, or more precisely, an agenda-driven execution model.
  - The *Construction Model (CM)* of DEMO is an abstraction that does not specify process, it provides just structural information.

## 5.4 Converting DEMO into BPMN

The goal is to convert the complete transaction axiom into BPMN, including all revoke types. Sections 5.4.1 to 5.4.4 describe all the necessary pieces and section 5.4.7 presents the result. We used BPMN 2.0 and leveraged the newly available *Data Store* construct.

[2]Apart from containing all the information from the other models.

### 5.4.1 C-acts

C-acts are essentially activities that take place in order specified by the transaction pattern. BPMN has the concept of *activities* and the order is specified by *sequence flows*. As C-acts are atomic, the appropriate activity type is *task*.

### 5.4.2 C-facts

As mentioned in section 5.1, a C-fact becomes existent in the world as a consequence of performing a C-act. Heller in his thesis [72] lists three possibilities of expressing C-facts using BPMN:

1. Not explicitly expressed – the existence of the fact-C is not explicitly expressed. It is indirectly realised by a sequence flow. This option is sufficient if revokes are not considered (see further).

2. Using a BPMN message – the actor, who performs the given C-act sends a BPMN message with the C-fact to the other actor (transaction participant).

3. Using a BPMN signal – the actor, who performs the given C-act emits a BPMN signal on creating a C-fact. This has the benefit that apart from the other actor, any other actor may subscribe to the signal reception, which is aligned with the PSI-theory, where facts are present in the world, not only in the transaction, thus available also outside the transaction (modeled by interstriction links).

However, under closer consideration, none of the above solutions are entirely sufficient for the correct handling of revokes. For each revoke act, the PSI-theory specifies a certain *state* in which the transaction must be. The state is formulated like "X or further": request(ed) or further, promise(d) or further, and so on. This is why we decided on another representation: the BPMN 2.0 *data store*, into which the state of the transaction is stored. This datastore is connected to every C-act activity by an association.

### 5.4.3 P-(F)acts

It is not necessary to store information about them creating a P-(f)act into the data store because they can be derived from C-(f)acts: According to the PSI-theory, the P-fact starts to exist based on acceptance of the product, so P-(f)acts can be expressed by an activity only. If need be (optimization of an implementation), they can be stored similarly to the C-(f)acts described above.

### 5.4.4 Actors

Swimlanes in BPMN are isomorphic to actors in DEMO [165]. BPMN lacks a higher abstraction level of actor roles, being the logical sum of responsibility, authority, and competence necessary to carry out the product [29]. There are generally two approaches:

(i) abstracting the swimlanes to actor roles (like Decider or Concluder), (ii) remaining on the BPMN's low level of abstraction, and using swimlanes to represent actors – company functional roles – like CEO or specific people like Jane.

Another possibility for representing actor roles is using BPMN *pools*, where each pool represents one actor. The resulting BPMN models will be very similar to models using swimlanes. However, we have not chosen this representation because: (i) The correspondence of actor roles and transactions is not explicit, (ii) sequence flow cannot be used between pools, which would result in using messages, further complicating the diagrams.

### 5.4.5 The Composition Axiom

A composition of transactions may be dealt with in two ways: (i) to model all the transactions in one diagram (ii), to the separate diagram for every transaction. Generally, both approaches are valid, but (ii) may lead to huge diagrams, as can be seen in fig. 5.8 and fig. 5.9. As (ii) guarantees the limit of the diagram size, we preferred it. On the other hand, it may make understanding the big picture harder.

We propose the following 2-part expression of the composite axiom:

1. **Launching a child transaction** in a specific place in the parent transaction. The child transaction must be started just after creating a specific C-fact. A *message-throwing event* may be used in case of initiating a single child transaction. In the case of firing multiple child transactions, signals are appropriate, similar to the C-acts above. Moreover, it is needed to ensure multiplicity. In case it is more significant than one, we need to initiate several child transaction instances. This is achieved either by using a *cycle* for creating child transactions or a *loop activity*. Modeling by cycle (fig. 5.2) means that the model contains an activity counting how many times the activity was run. After this activity, there is a gateway. If the counter has not reached the number of child transaction instances to spawn, the process goes into a message throwing event to start a child transaction instance, and then the process returns to the counting activity. This happens $0..N$ times, as required. When multiplicity is modeled by a loop activity (fig. 5.3), the activity is in the form of a subprocess (with parallel loop), which sends a signal[3] that starts a child transaction. In the examples described below, the first (counter) variant is used because the model is more explicit. At the same time, for models with a multitude of child transactions, the more concise loop variant is recommended. Also, from the execution point, the implementation variant may be driven by the vendor, as a correlation of instances must be ensured (more discussed in section 5.4.8).

2. **Blocking execution of the parent process** until the child process has not reached the given state (creating a C-fact being waited on). This blocking can be realized by a BPMN *catching event condition* in the parent process, waiting for a specific condition before the given C-act. Here, a conditional event must be used instead of a

[3]We cannot use a message send in this situation, because the encapsulation would be violated.

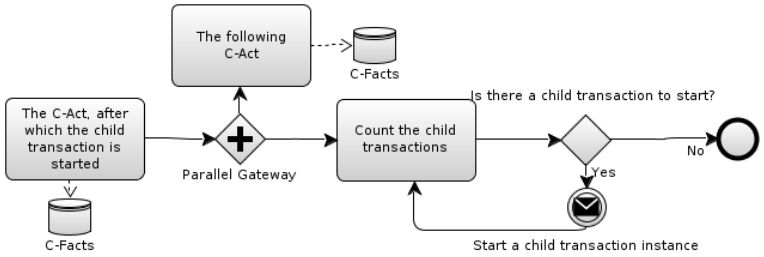


Figure 5.2: Launching child transactions by using counter

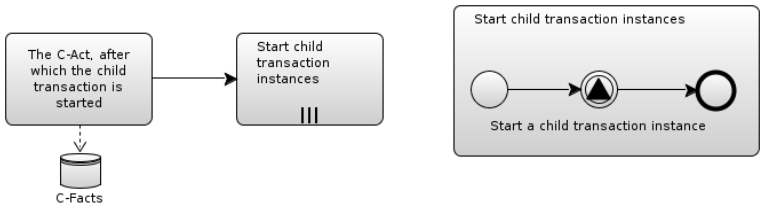


Figure 5.3: Launching child transactions by using loop

signal event, as we do not wait just for a signal but for a specific instance in the case of multiple child transaction instances. This situation is modeled in fig. 5.4. Again, specific vendor correlation techniques may apply (section 5.4.8).

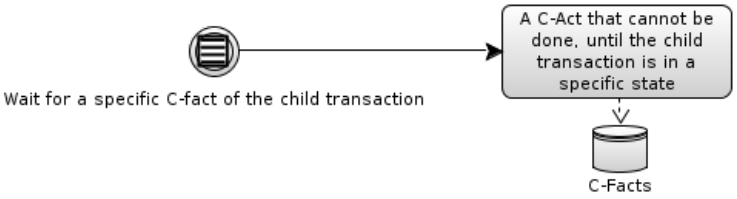


Figure 5.4: Waiting for a child transaction

### 5.4.6 Revokes

Revokes are the most challenging part of the conversion. Let us present the challenges and how we dealt with them:

- A revoke must be applied on a specific instance of the transaction; in a certain time, there can be several parallel transaction instances running. This must be ensured by the BPMN system (section 5.4.8).
- A revoke can be fired independently on the running main process. It can be modeled straightforward, as BPMN allows several independent start events.
- A revoke can be fired only if the transaction is in an allowed state. This we ensure by an activity checking the state of the transaction, which was previously stored into a data store.

- When revoking a C-fact, after which a child transaction has been started, the child transaction must be completely revoked. This is done by calling a compensation throwing event by the revoke, followed by performing the compensation activity by the corresponding parent transaction.

- In the process flow, there can happen a situation that a P-fact was already created (the P-act has been finished), while a revoke moves the process to a state preceding performing the P-act. In this case, it is necessary to "throw away" the P-fact. We solve this using a BPMN compensation element and the respective compensation activity, similarly to the previous point.

- A revoke must be initiated by the actor who performed the respective C-act to be revoked. This is ensured by using the same identifier for the swimlane of the actor role initiating the revoke as for the actor role of the respective transaction.

A revoke works in the following steps according to the transaction pattern. First, the revoking actor asks the other actor to grant the revoke. The other actor allows or refuses. If the revoke is allowed, the main process returns to the appropriate state. We model this by using simple BPMN *subprocess* with a set of appropriate activities (fig. 5.6).

### 5.4.7 The Resulting BPMN Model

The complete transaction pattern described by the BPMN notation illustrates fig. 5.5[4]. Although it describes only one transaction, it is very complex and complicated. As it is presented in section 5.5 and discussed in section 5.6, models containing more than one transaction are not easily readable by usual readers, and it is recommended to use them for the process execution in BPM systems.

### 5.4.8 The Execution

Apart from documentation purposes, BPMN models can be simulated and/or executed. While designing the conversion, we tried to make the resulting BPMN model precisely following the required behavior. Unfortunately, the BPMN standard does not specify the execution implementation details. Each company developing a BPM system (system for modeling, simulation, and execution of processes), as Intalio, BizAgi, or IBM, has their specific implementation, which requires various additional modeling and programming steps necessary to make the model executable. At the same time, some of the BPMN constructs may not be supported, or they are implemented differently. All these aspects must be taken into consideration for turning the resulting BPMN models into an executable form. Generally, here are the things that must be implemented:

[4]This and the following models may not be legible in the printed version. We recommend obtaining the electronic (zoomable) version. The source models may be downloaded from `https://ccmi.fit.cvut.cz/methodologies/bpmn/`

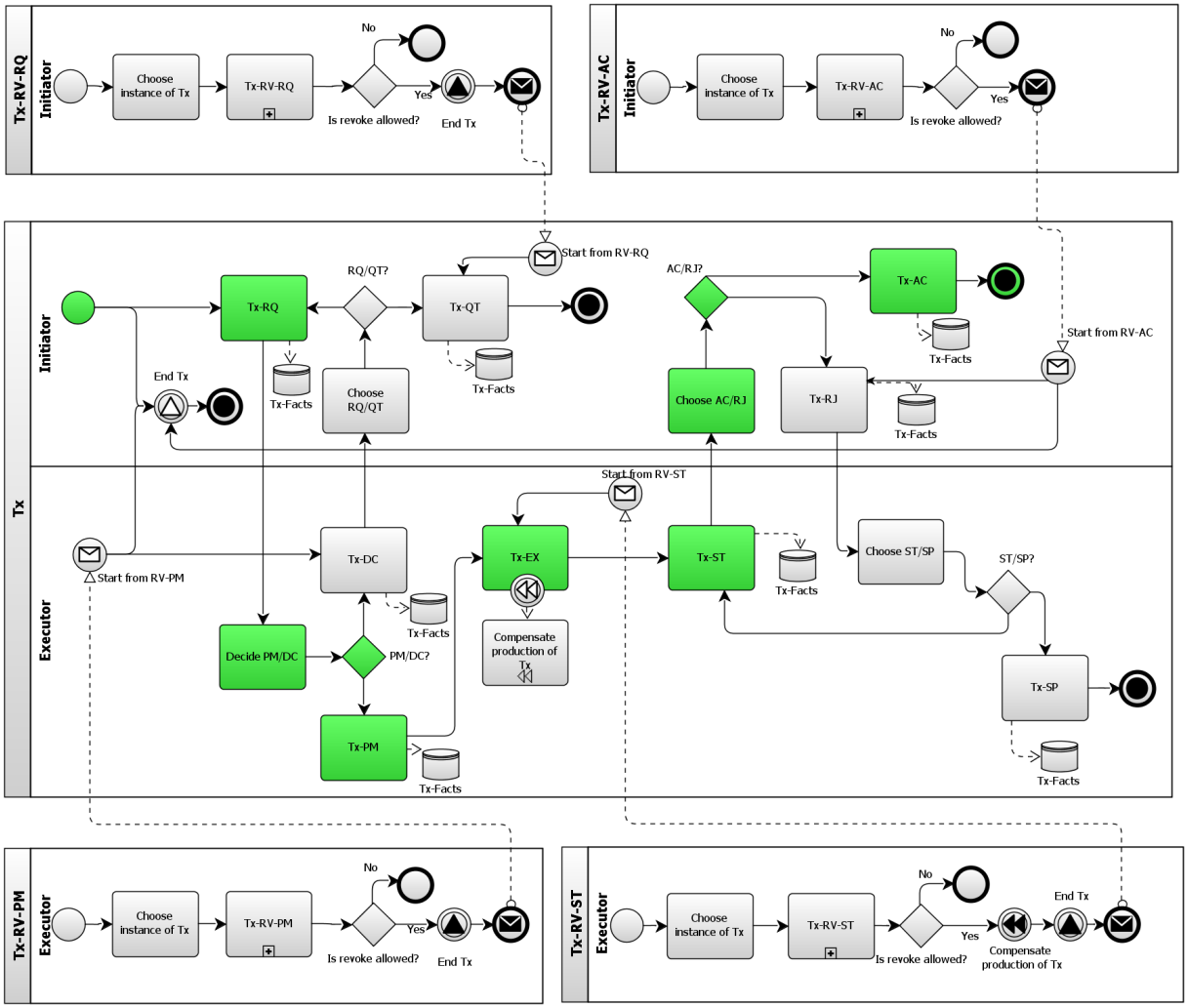


Figure 5.5: Transaction in BPMN, happy flow is marked by green colour

- Agenda handling. The possibility to start a process and provide a "task inbox" of the required reactions on the originating C-facts. This requires developing some sort of user interface (UI).

- Allowing the participants to make their choices. Again, some sort of UI solves this. Also, some choices may be determined by complex facts evaluation specified in the Action Model. There are two possible approaches:

  1. Leaving the evaluation to users, which means the users have the rules in their head or consult the Action Model or any other codification of the rules.
  2. Programming the BPM system to (help) evaluate the rules. The extent to which the automation may happen depends on the BPM system's possibilities and on the context (the availability of the necessary data in the company's technological systems and their accessibility).

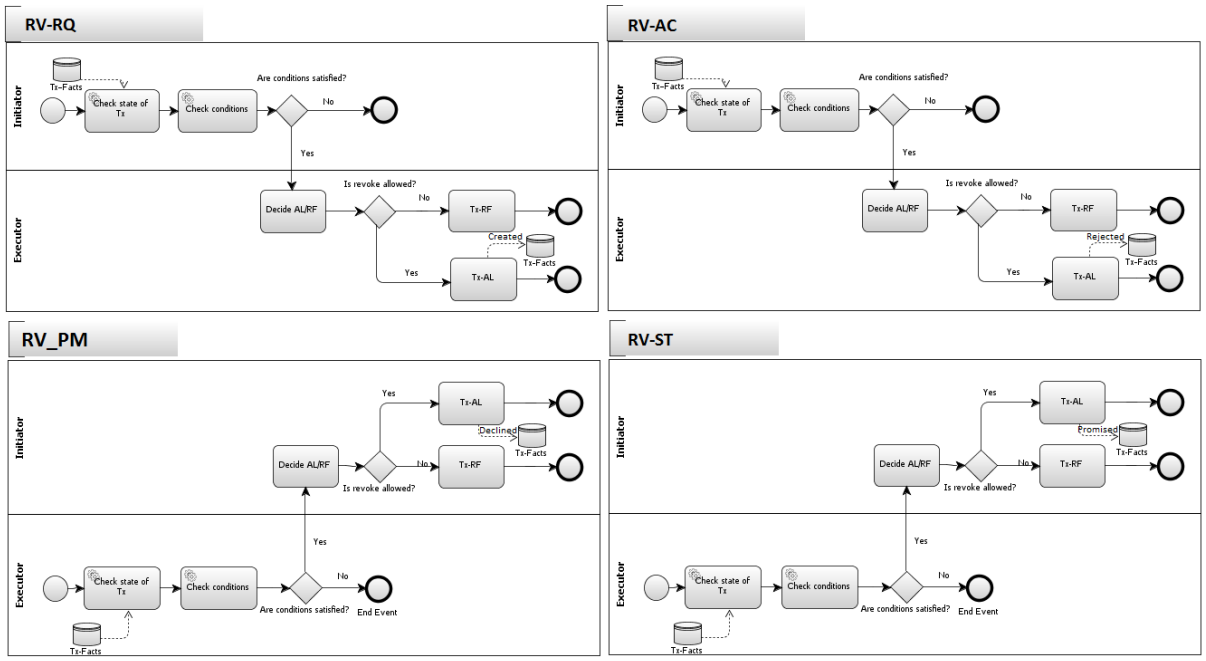


Figure 5.6: Revokes in BPMN

- Signals handling.
- Implementation of reading and writing data to data stores.
- Instance matching. Specific instances of transactions must be matched in some situations as child transactions (section 5.4.5) and revokes (section 5.4.6). Intalio and Oracle call this concept a "correlation".

## 5.5 Example – Case Voley

As an example for the demonstration of our method, the traditional Case Voley example [33] was selected because of its simplicity, yet including the substantial constructs. In fig. 5.7 there is the OCD diagram of this example.

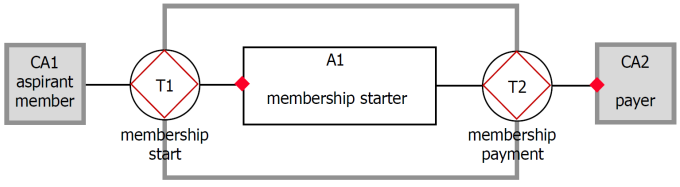


Figure 5.7: OCD of Case Voley [33]

The process has two transactions and three actors. The transformed BPMN model converted by the described method is in fig. 5.8 and fig. 5.9 . Subprocesses depicted in fig. 5.6 are not shown here, as they are generally the same.

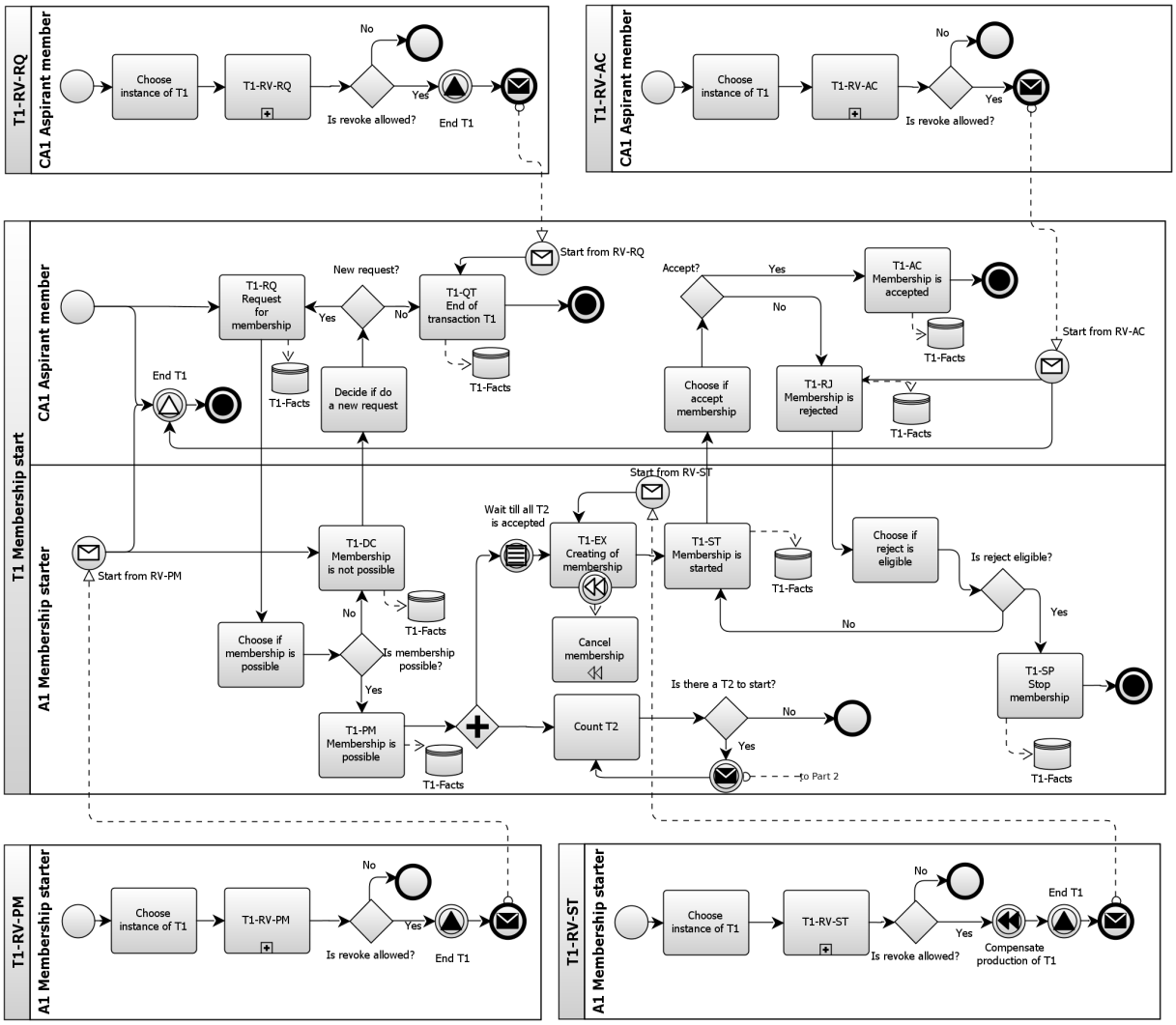


Figure 5.8: Case Voley converted into BPMN – part 1

## 5.6 Discussion and Conclusions

The limitation of typical BPMN models from the view of the PSI theory lies in their limited expression of reactions to unexpected situations. Many situations like decline, reject, and especially revokes are not covered in the models, which causes operation troubles. The presented conversion method offers a remedy to this by bringing the *complete transaction pattern into BPMN*, which means including all revokes. Moreover, compared to the previous efforts, our method deals with the spawning of *multiple child transaction instances*

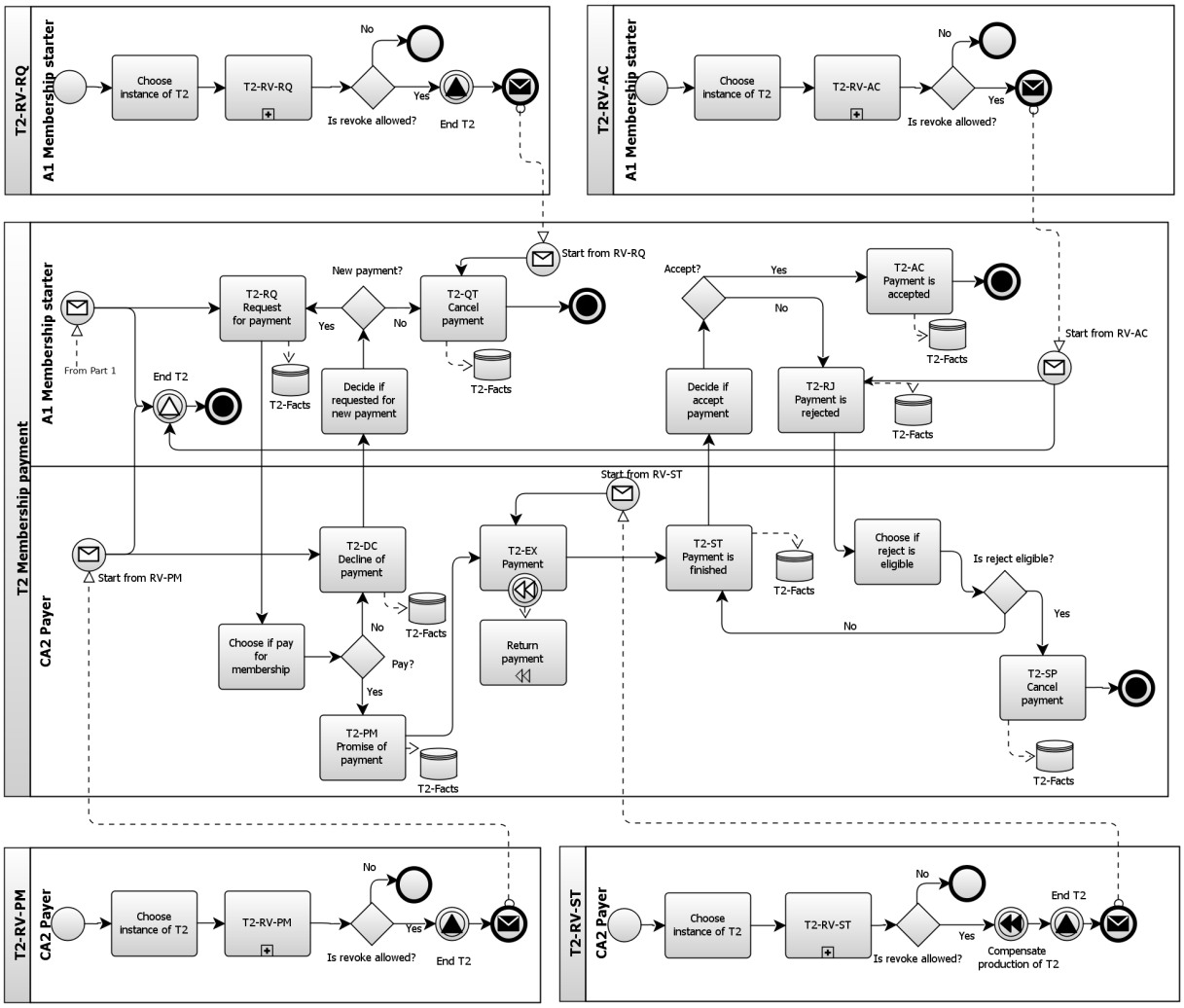


Figure 5.9: Case Voley converted into BPMN – part 2

(initiation links with multiplicity $\neq 1$) and waiting for them in the parent transaction (waiting links with multiplicity $\neq 1$). Also, the resulting models are *executable*.

As for the DEMO models covered, the described conversion method covers the Construction Model plus the Process Model. Based on a concrete BPM system implementation, decision rules contained in the Action Model can be incorporated in the respective activities, as described in section 5.4.8, which is also true for rules from the Fact Model.

The concept of interstriction has not been discussed. However, a keen reader has probably realized that whenever an actor in its activity needs specific information from another transaction, it is simply modeled by accessing the respective transaction data store.

The example shows that in spite of the simplicity of the DEMO model involved, the resulting BPMN model is complex. The reason is mostly the complete transaction pattern, which covers all the possible situations according to the PSI theory. The question arises about human readability. There are several points to this topic:

1. In practice, the model may be made smaller by leaving out the parts which are not applicable (which means they (almost) never happen). These are typically the revoke patterns.

2. Yet, for complex models, the resulting size may remain still unmanageable. In this case, it would be advisable to cut the model into smaller pieces using some sort of decomposition and/or *link* BPMN elements. The concrete way to do this may be explored in future research.

3. It is questionable whether human readability is required. If one wants human-readable diagrams according to the PSI theory, the DEMO diagrams are the solution, as they have been tailored to it. It may be the case that learning and applying them comes at a lower cost than forcing the diagrams into a BPMN notation, just because "BPMN is the standard".

Our stance is that the most significant possibilities of our method lie in *machine readability*, which means generating BPMN models that can be fed into a BPMN execution system to implement an automated workflow that is able to react to every possible situation specified by the complete transaction pattern, not just a typical BPMN "happy path with a bit of branching".

Apart from converting the DEMO models, the conversion may also be applied for analysis of existing BPMN models of business processes as described in section 5.2.1. The way of working would be to transform the BPMN models into DEMO and then generate the "supercharged" BPMN version by converting them back using our method.

As for future work, verification of bigger models from practice is necessary. As such conversion will not be feasible by hand, an implementation of the conversion automation will be required.

**Acknowledgements** This research has been funded by CTU SGS grant No. SGS16/120/OHK3/1T/18. The authors wish to sincerely thank ForMetis BV and especially Dr. Steven van Kervel for the kind support of this research.

## 5.7 Chapter Summary

In this chapter, we proposed a method to transform DEMO models into executable BPMN models. The method was demonstrated in a Volley club example. The evaluation of the related research question and DSR cycle is discussed in Chapter 9.

Chapter 6

# An Experiment in the Procedural Law Domain – 32 Case Studies

This chapter focuses on empirical case studies for the main centralized compliance DSR cycles based on the research objective. Section 6.1 describes the experiment design. Section 6.2 presents a representative case study for demonstration purposes, and Section 6.3 overviews all case studies. Then, Section 6.4 discusses the results. Finally, Section 6.5 summarizes the results.

## 6.1 Experiment Design

An experiment was designed to answer the research objective from Section 1.4 by applying the method described in Section 6.1.1 to real-world process descriptions in the domain of Czech procedural law. The research question for the experiment was formulated as: "Can EE theories be used to increase the quality of business process requirements for process-based information systems?" The legal domain was selected because legal texts are inherently written in a detailed and minimally ambiguous manner. In addition, gathering sufficient software specification requirement texts for research is challenging, as they are subject to company privacy. The case studies may be considered a showcase of the semantic challenges of the full digitalization of procedural law.

The experiment comprised the following goals:

1. To demonstrate the feasibility of applying DEMO methodology to produce formal models with C4E qualities in the domain of procedural law based on the legal descriptions.

2. To study the potential of this approach in increasing the quality of the software requirements; to observe and evaluate how the method guides a modeler towards reducing the semantic ambiguity of the domain description to answer the research question.

The case studies were modeled based on publicly available legal texts, without consulting legal domain experts to assess the clarity and ambiguity of the source documents. The study was conducted as a part of a master-level business process modeling course at the Czech Technical University in Prague over three years, where students modeled parts of the legal procedures.

After completing the DEMO models, the students designed executable BPMN models as described in Section 6.1.1 and executed them in a BPM engine to create a prototype of the information system supporting procedural law.

In the first year, the first proof-of-concept example was created based on a dispute settlement procedure from the Czech Arbitration Court [96]. This was accomplished in close cooperation with the lecturers, and a reference example of the application of the method was completed. In the second year, a template was created for the students based on it, and they worked in teams of two, according to the steps described below [144]. In the third year, the students worked individually and had access to the previous year's cases [143]. The experiment was conducted according to the following steps:

1. Domain descriptions from the Czech procedural law that contain complex process description were identified and prepared. The length of the domain description was approximately 2 000 words on average.

2. Students were trained in DEMO modeling for approximately 40 h.

3. Students worked on the case studies according to a description of the method and reference Arbitration Court case study. They were given a template based on the proposed method.

4. Students presented progress of their work to their peers and lecturers and received feedback to improve the models.

5. After a final submission, the case studies were evaluated by lecturers and only the correct case studies were included in the final dataset; 23 of 32 works (68.75%) were included in the published dataset.

In total, the case studies contained 115 276 words of legal text. The case studies selected for the final dataset contained 85 421 words. Measuring the exact duration for individual case studies was beyond the scope of this experiment. Student cases were limited by the scope of the class, which was set to 70 h. The Arbitration Court reference case study required approximately 200 h. Therefore, all case studies required approximately 2 440 h to model, and the case studies selected for the final dataset required 1 740 h.

The key limitations of the case study are as follows.

- Owing to the exclusion of some case studies caused by a low quality of work accomplished by some students, the remaining case studies cover only some parts of the legal text. This does not affect the experiment, as the text contains many different

processes, and the exclusion of some processes does not affect the evaluation related to the research question.

- The case studies were not elaborated upon using multiple approaches for comparison. First, the study's human resources were limited, and second, no other suitable, similarly rigorous approach for identifying missing information in process descriptions has been identified (see Section 6.4.5).
- The proof-of-concept implementations in Camunda BPM system were created only for a subset of the DEMO model process specification. However, as this study is unconcerned with the technical implementation, this also does not affect the results.
- Except the reference case study, all studies were performed on a domain description in the Czech language; thus, the reproducibility of the experiment is limited by the language.
- Only three of four DEMO aspect models were elaborated. The AM was excluded because it is too detailed and time consuming for the limited human resources of this study.

### 6.1.1 Method Overview

Figure 6.1 presents a graphical overview of the proposed method.

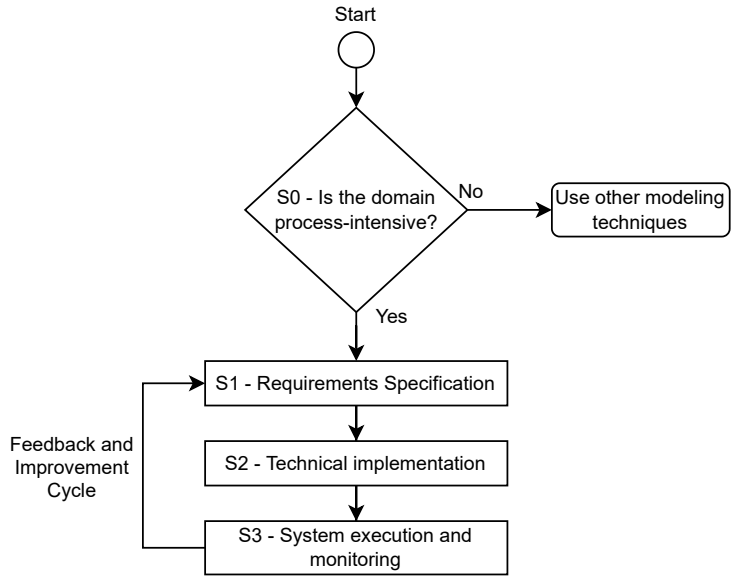


Figure 6.1: Conceptual overview of the proposed method

*S0 – Is the domain process-intensive?* – This step involves evaluating the domain to determine whether it contains complex processes to be implemented. The proposed method applies only to business processes involving human cooperation and co-production

because it is based on Habermas's sociological theory of communicative acts [63]. If the target information system is not expected to be controlled by human actors, using modeling languages described in Section 2.2.1 and Section 2.2.2 is preferable. The method works with any complexity and size of processes; however, the greatest benefits are expected for large process descriptions typically found in large companies. This method is suitable for the case management process mentioned in Section 2.2.2.

*S1 – Requirements Specification* – This step involves applying the DEMO OER method to a process from a target domain and creating DEMO aspect models. This is expected to guide modelers to produce better-quality process models in software requirements for a target software system. This section provides a detailed description of this, starting from Section 2.3.3. Section 6.1.1 then discusses the applications instructions for 23 case studies to evaluate the benefits of this approach.

*S2 – Technical Implementation* – This step involves transforming the process models into a conceptual language used by the selected BPM system or a low-code platform. Providing detailed guidance is outside the scope of this study; the research question is limited to the quality of process descriptions in software requirements. However, the 23 case studies presented include proof-of-concept implementations in the BPM system Camunda [17].

Because technical requirements for target IT systems significantly vary, universal steps for implementing DEMO-based software requirements are difficult to formulate in general. However, a formal execution language called DEMO Machine [A.3] has been established and can be used as a starting point to guide the implementation of any system. Furthermore, [A.2] presents a blueprint for transforming DEMO into BPMN. It presents a method for a complete transaction pattern transformation; however, in practice, BPMN models must typically be simplified to avoid size explosion, which is difficult to automate. For low-code platforms, [91] provides a description of how to translate DEMO models into a Mendix platform.

Steps *S3 – Monitoring and Execution* and *Feedback and Improvement Cycle* are outside the scope of this study and are addressed according to the best practices for BPM, such as [166].

Next, we introduce the basic concepts of PSI theory (thinking approach) to enable us to describe how they are used in the DEMO method (working approach).

## 6.2 Reference Example – Arbitration Court

In this section, we present a reference example that is the basis for conducting the case studies. To understand the DEMO models presented in this section, having a solid understanding of the method (Section 6.1.1) is advisable, particularly Section 2.3.5, which provides guidance on reading DEMO CMs.

Arbitral proceedings are alternatives to classical court proceedings. They are decided by one or more arbitrators, who are experts on the topic of the dispute. The proceedings lead to the rendering of the arbitral award, a legally binding final decision on the dispute.

The Arbitration Rules of the International Arbitration Court of the Czech Commodity Exchange [127] have two official language versions, Czech and English. The rules define the procedure of conducting the arbitral proceedings and all actors involved. This process is governed by Act No. 216/1994 Coll., on arbitral proceedings and enforcement of arbitral awards. A complete case study is available from GitHub [96].

#### 6.2.0.1 Organization Essence Revealing

The first step of the OER method (as explained in Section 2.3.3) involves going through the text, identifying the products, highlighting the communication acts, and determining whether the acts are ontological, infological, or datalogical. In this case, only ontological transactions were modeled. The legal documents used had 41 pages of text; therefore, only a brief example of this process is provided.

The following paragraph is Article 7, paragraph 3 of the rules; it is part of the process of challenging an arbitrator, as discussed in Article 7. Ontological transactions are underlined with a solid line, and infological transactions are indicated by a dashed line. The subjects participating in the transactions are marked with dotted lines.

> If the challenged arbitrator, having been informed of the challenge[request,ontological], considers the challenge groundless and does not resign from his or her office[decline,ontological], the Presidium of the Arbitration Court is authorized to decide on the challenge[state,ontological]. The Presidium of the Arbitration Court assesses the admissibility of the challenge in terms of paragraphs (1) and (2), and if the Presidium concludes that the challenge was made properly and in time[decline,ontological], the Presidium decides on the merits of the challenge[state,ontological]. The Presidium of the Arbitration Court provides the challenged arbitrator, the remaining members of the arbitral tribunal and the other party or parties with an opportunity to comment on the challenge[infological] before any decision is made thereon. Any and all of the above-mentioned statements will be communicated[infological] to all parties and arbitrators. The Presidium of the Arbitration Court may decide that the parties shall not have access to a statement and/or a part thereof[infological] provided by any arbitrator if it contains inside information regarding the actions of the arbitral tribunal relating to the proceedings and the factual and legal assessment of the case.

The second step of the OER method identifies transactions, products, actor roles, and organizational roles. For each ontological communication act (underlined text) from the previous step, the modeler identifies its product and places it in the extended transaction result table (Table 6.1). The transaction name was derived from the product name. Therefore, if two communication acts share the same product, they belong to the same transaction.

Organizational roles (dotted text) are typically mentioned together with communication acts. However, they are not directly associated with transactions because different acts

Table 6.1: One transaction from the extended transaction result table [96]

| **Transaction** | **Challenging an Arbitrator (T13.1)** |
|---|---|
| Product | Challenge of an Arbitrator is Resolved (P13.1) |
| Initiator | Arbitrator Complaint Completer (A13) (Identified in Step 3) |
| Executor | Challenge Arbitrator Completer (A13.1) |
| Request | Filing Challenge of Arbitrator (§7/2) |
| Promise | Not Specified |
| Decline | Not Admitting the Challenge (§7/2) |
| Declare | Challenge Resolved |
| Reject | Not Specified |
| Accept | Not Specified |
| Revoke Request | Not Specified |
| Revoke Promise | Not Specified |
| Revoke Declare | Not Specified |
| Revoke Accept | Not Specified |

of a transaction can be performed by different organizational roles. The executor of a transaction is derived from the transaction product. The initiator is either derived from the product or is an executor of the parent transaction identified in step 3. The mapping between the organization and DEMO actor roles is performed in a subject actor table that is excluded in this section for simplicity.

The second step of the OER method must often be performed multiple times because some transactions are discovered to share the same product and can be merged. In this case study, the authors began with 120 transactions and ended with 37.

The final step of the OER method identifies that almost all transactions belong to the same dispute-settlement process. The parent transaction is arbitral proceedings (T1), and Figure 6.2 shows the transaction tree it forms. Further nesting is indicated by the transaction identifier with the pattern `[parent.child]`. The second process tree consists of only one transaction, T14, which reviews the arbitral award. It was modeled as a separate process because it occurs after the arbitral proceedings are performed.

#### 6.2.0.2 Modeling Organizational Essence

The construction model was created for all 37 identified transactions and divided into four parts for better comprehensibility. Figure 6.2 provides the main overview of the process, which begins with T1 (arbitral proceedings) and comprises T2-T13. CA5, CA6, and CA13 were modeled as composite actors and exhibited the encapsulation of further process nesting. Figure 6.3 illustrates this, which is an expanded composite actor CA5 from a high-level construction model. The complete construction model is available from GitHub [96].

For convenience, simplified PM information was included in the CM in the form of cardinalities and conditional links (dashed lines). The cardinalities indicate the number of child transactions that can be created by a transaction type. When the cardinality begins at zero, the transaction is optional. Conditional links mean that the destination executor's transaction cannot be executed before the source transaction is executed and accepted. For example, T3 cannot begin before T2 is accepted. Note that all transaction instances can run in parallel and are synchronized by conditional links. This means that T10 can progress while T3 waits for the completion of T02.

## 6.3 Case Studies Overview

All the case studies are summarized in Table 6.2 and published on GitHub [96, 144, 143].

## 6.4 Evaluation and Discussion

In this section, the results of the experiment presented in the previous section are evaluated with respect to the research question and experimental goals presented in Section 6.4.1 and Section 6.4.2. The implications of this research are suggested in Section 6.4.3. Section 6.4.4 discusses the limitations of the proposed approach. Section 6.4.5 compares similar approaches with ours. Finally, Section 6.4.6 proposes possible future research topics.

### 6.4.1 Goal 1 – Feasibility

Goal 1 was formulated in Section 6.1 as "To demonstrate the feasibility of applying DEMO methodology to produce formal models with C4E qualities in the domain of procedural law based on the legal descriptions." Of the 32 case studies in the experiment, 23 were modelled successfully, yielding a 68.75% success rate. After a closer examination of the remaining cases, the failure was concluded to result from the students' lack of competence, and they could have been modelled successfully by a competent modeler. The reason for the failure to deliver a quality model may be the short training period (40 h) or the students' insufficient commitment to the task.

Therefore, we can conclude that all case studies in the experiment can be modelled using the proposed method and that applying the DEMO methodology to produce formal models with C4E qualities in the domain of procedural law based on legal descriptions is possible. The C4E qualities are guaranteed by the applied DEMO methodology, as explained in Section 2.3.4.

After capturing the process-based domain requirements, a proof-of-concept implementation was conducted in the BPM system Camunda. This demonstrated the feasibility of using our approach together with state-of-the-art BPM systems and low-code platforms, which are typically based on BPMN or its subset. As the details of this are beyond the scope of this study, we do not elaborate on it further.

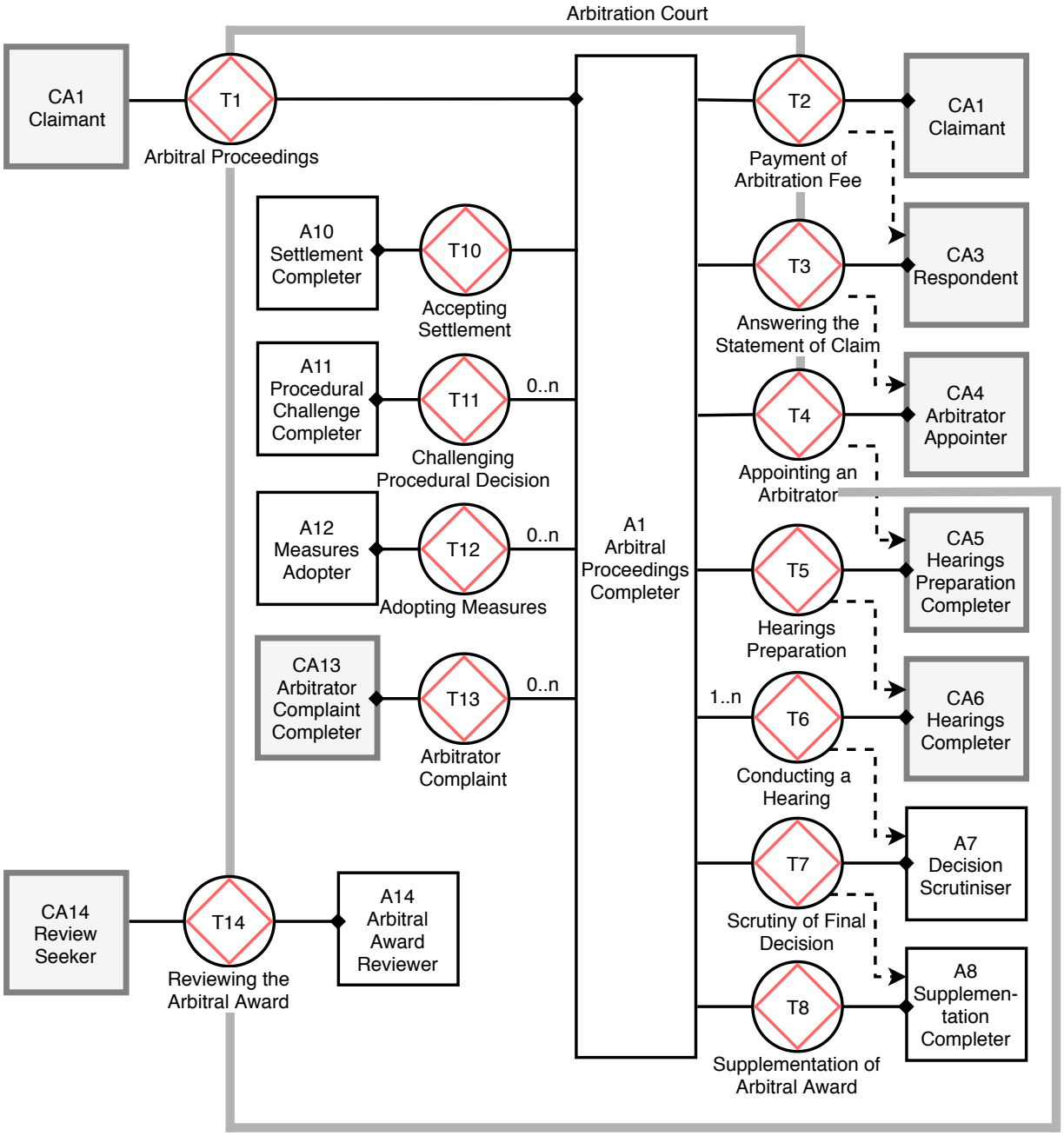


Figure 6.2: High-level construction model

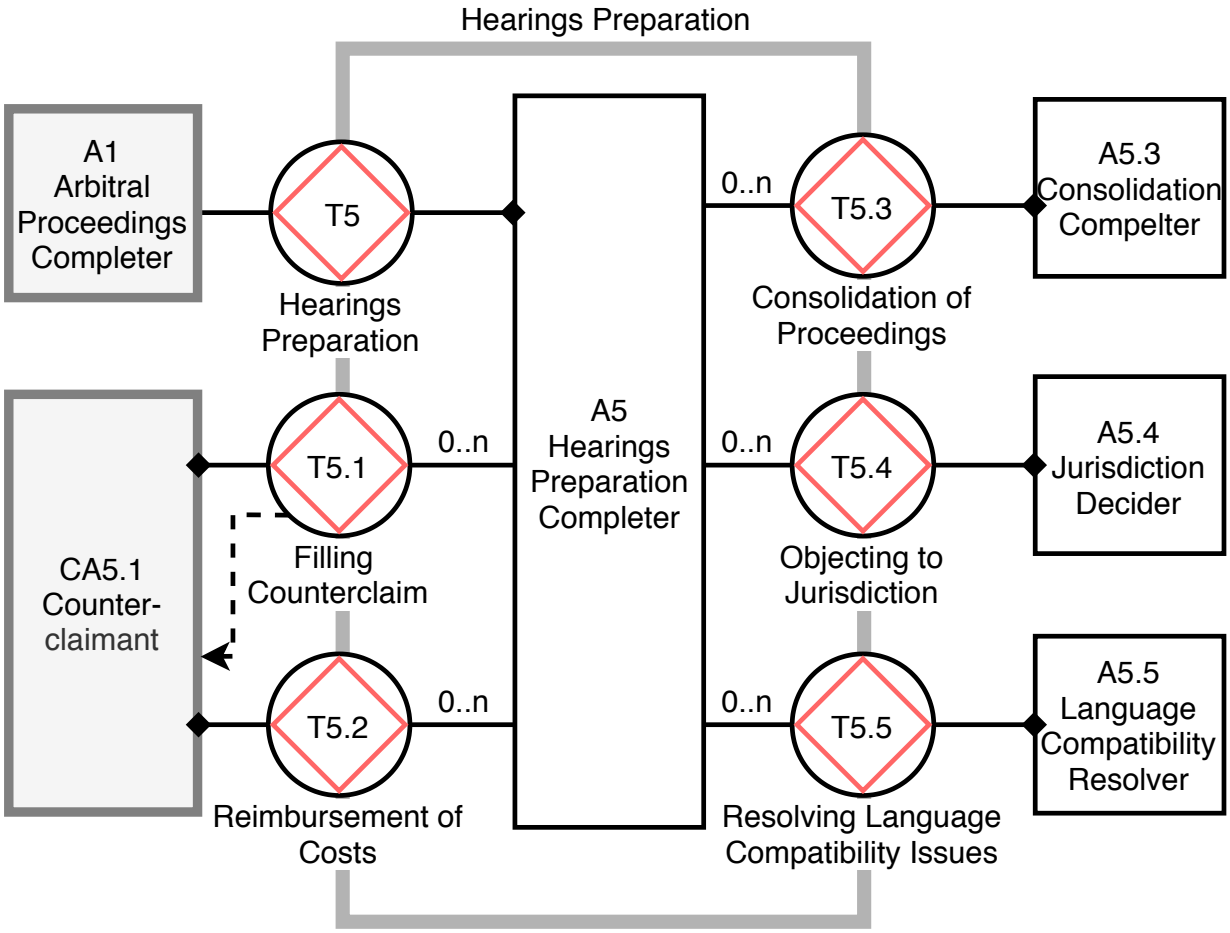


Figure 6.3: Construction model of the hearings preparation

### 6.4.2 Goal 2 – Increasing the Quality of Software Requirements

Goal 2 was formulated in Section 6.1 as "To study the potential of this approach in increasing the quality of the software requirements; to observe and evaluate how the method guides a modeler towards reducing the semantic ambiguity of the domain description to answer the research question".

Based on the results, we can formulate several observations in which the presented method can lead to a quality increase in:

1. identifying missing coordination acts;
2. identifying missing actor roles; and
3. managing complexity.

Table 6.2: Case studies overview and discovered acts (N=23)
(AC=Arbitration Court, CCP = Code of Civil Procedure, CP = Criminal Procedure, TR = Tax Regulations)

| Case Study | Transactions | Request | Promise | Decline | Declare | Reject | Accept | Revoke Request | Revoke Promise | Revoke Declare | Revoke Accept |
|---|---|---|---|---|---|---|---|---|---|---|---|
| AC | 37 | 32 | 3 | 10 | 37 | 1 | 0 | 3 | 1 | 0 | 1 |
| CCP 3.4 | 17 | 14 | 0 | 14 | 15 | 0 | 0 | 3 | 0 | 0 | 0 |
| CCP 6.1 | 10 | 9 | 1 | 2 | 0 | 1 | 0 | 4 | 0 | 0 | 0 |
| CCP 4.2+4.3 | 14 | 4 | 2 | 5 | 8 | 5 | 3 | 1 | 0 | 1 | 1 |
| CCP 4.1 | 16 | 2 | 0 | 1 | 15 | 1 | 0 | 1 | 0 | 0 | 0 |
| CCP 6.5.2 | 29 | 16 | 16 | 4 | 3 | 0 | 0 | 1 | 0 | 2 | 0 |
| CCP 3.2+3.3 | 28 | 21 | 10 | 1 | 10 | 3 | 1 | 0 | 6 | 0 | 0 |
| CCP 3.1.3 | 23 | 22 | 2 | 2 | 3 | 1 | 5 | 1 | 0 | 0 | 0 |
| CCP 3.1.1 | 9 | 7 | 6 | 6 | 2 | 0 | 0 | 0 | 1 | 0 | 2 |
| CP 3.18+3.19 | 7 | 3 | 2 | 2 | 2 | 0 | 0 | 2 | 0 | 0 | 0 |
| CCP 5+6.6 | 10 | 1 | 1 | 3 | 8 | 0 | 2 | 1 | 0 | 0 | 0 |
| CCP 6.5.2 | 20 | 18 | 14 | 10 | 13 | 4 | 11 | 2 | 5 | 2 | 0 |
| CCP 2.8+9+10 | 14 | 13 | 3 | 6 | 11 | 1 | 2 | 1 | 3 | 0 | 0 |
| TR 2.6a | 16 | 8 | 4 | 5 | 12 | 4 | 5 | 2 | 0 | 1 | 1 |
| TR 2.6b | 11 | 7 | 1 | 3 | 7 | 3 | 3 | 1 | 0 | 1 | 0 |
| TR 3.1+2+3+4 | 12 | 8 | 5 | 1 | 8 | 1 | 0 | 3 | 1 | 5 | 2 |
| TR 3.5a | 11 | 11 | 10 | 5 | 7 | 5 | 1 | 1 | 1 | 0 | 0 |
| TR 3.5b | 21 | 9 | 3 | 1 | 3 | 3 | 2 | 4 | 2 | 3 | 0 |
| CP 2.9 | 27 | 17 | 8 | 3 | 11 | 2 | 6 | 5 | 2 | 0 | 0 |
| CP 2.10a | 11 | 4 | 3 | 4 | 9 | 5 | 2 | 2 | 2 | 3 | 0 |
| CP 2.10b | 14 | 14 | 5 | 2 | 2 | 1 | 1 | 1 | 0 | 0 | 0 |
| CP 3.13 | 11 | 9 | 1 | 4 | 7 | 2 | 1 | 1 | 1 | 0 | 0 |
| CP 3.13+14 | 23 | 20 | 11 | 3 | 19 | 8 | 4 | 0 | 2 | 5 | 1 |
| **Total** | **354** | **237** | **108** | **87** | **175** | **50** | **49** | **37** | **26** | **23** | **7** |

#### 6.4.2.1 Identification of Missing Coordination Acts

DEMO specifically defines each possible step of a transaction using the complete transaction pattern (Figure 2.4) based on Habermas's theory of communicative acts [63]. Its completeness has been proven empirically; 20 years of applying DEMO in practice have not discovered fundamental missing pieces. Although suggestions for their extension exist [58], they have not been determined to be crucial by the EE community.

In the presented case studies, all transactions had missing definitions of several coordi-

nation acts. The transactions were typically specified in one sentence describing the actors and transaction results. Table 6.3 presents the missing process steps aggregated from all 23 case studies, and Figure 6.4 shows a plot based on this table. In the experiment, 80.10% of the information related to the coordination acts was missing.

Table 6.3: Missing coordination acts in all case studies (N=23)

| | Specified | Not Specified | Missing Information |
|---|---|---|---|
| Standard Transaction Pattern | | | |
| Request | 237 | 117 | 30,05% |
| Promise | 108 | 246 | 69,49% |
| Decline | 87 | 267 | 75,42% |
| Declare | 175 | 179 | 50,56% |
| Reject | 50 | 304 | 85,88% |
| Accept | 49 | 305 | 86,16% |
| **Total** | **706** | **1418** | **66,76%** |
| Revokes | | | |
| Revoke Request | 37 | 317 | 89,55% |
| Revoke Promise | 26 | 328 | 92,66% |
| Revoke Declare | 23 | 331 | 93,50% |
| Revoke Accept | 7 | 347 | 98,02% |
| **Total** | **93** | **1323** | **93,43%** |
| Complete Transaction Pattern | | | |
| **Total** | **799** | **2741** | **80,10%** |

The least numbers of missing steps were for *request* and *declare*. This is expected, as they represent the beginning of the process and delivery of the resulting product; they are not typically forgotten, although surprisingly high amount of vagueness remains even in these steps (30.05% and 50.56%). The second-least numbers of missing steps were for *promise* and *decline*, that is, confirming or refusing the actor's request; the *accept* and *reject* coordination steps enacted upon the delivered transaction product were missing in 86% cases.

The largest number of missing steps were for *revoke* acts, which represent withdrawing previous commitments of the actors; these were not present in 93.43%. This is understandable, as "going back" is a relatively drastic measure that may be forbidden, particularly in legal processes. However, we argue that the specification should explicitly mention these; otherwise, whether revokes are completely forbidden is unclear, or even whether situations in which they can be handled exist (although rare).

However, even when disregarding revokes, that is, following only the simplified Standard Transaction Pattern, 66.76% of information regarding coordination acts was missing, as shown in Table 6.4. Tacit coordination acts (not explicitly mentioned in the text) are undesirable and represent the missing information. In such cases, these steps are generally assumed to be performed without being expressed directly. Such a situation is exceed-

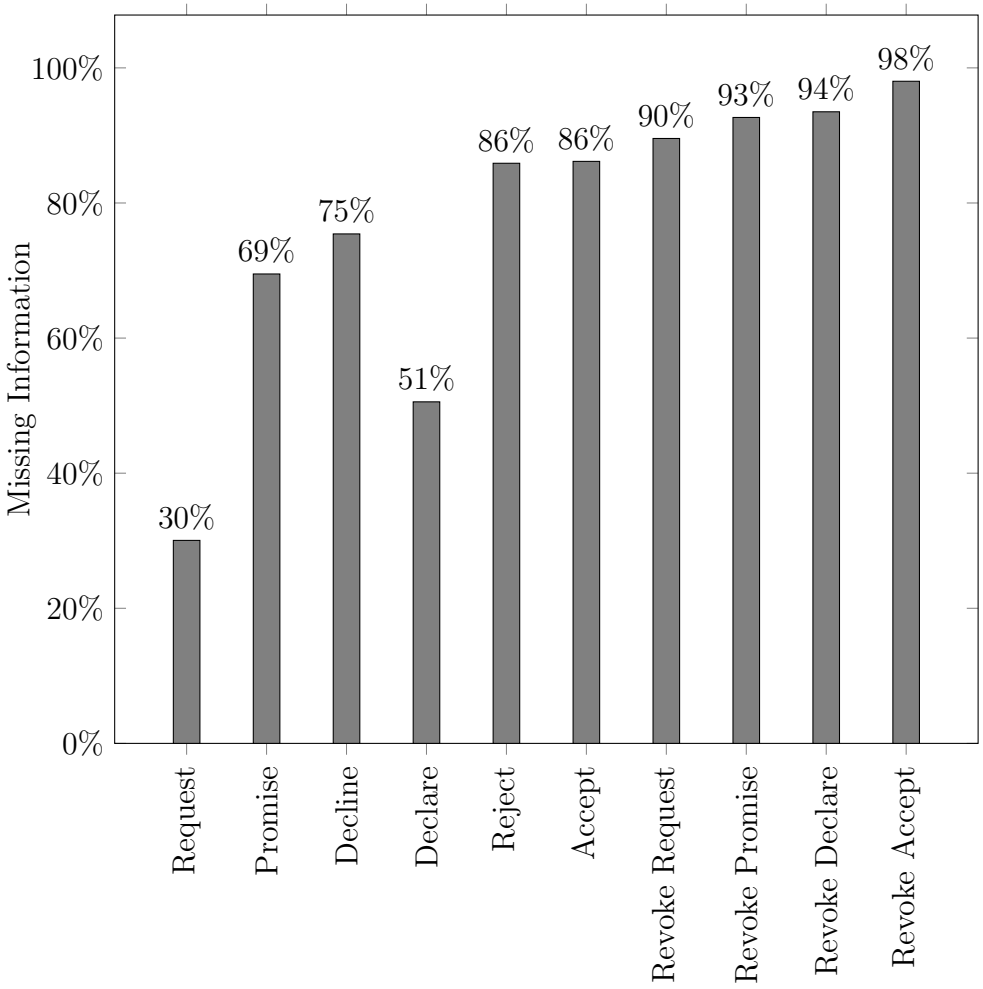


Figure 6.4: Missing coordination acts in all case studies (N=23)

ingly common in everyday processes; however, it poses the risk of misunderstanding and, therefore, should be avoided in software requirements. After further investigating some of these situations in the case studies, the missing transaction steps were determined to have a clear definition in the domain, but it was missing in the source material. Therefore, **we can conclude that the applied method has a high potential for discovering missing coordination acts in the specification, which contributes to increasing its quality**.

#### 6.4.2.2 Identification of Missing Actor Roles

Another benefit of applying the method to the domain specification is the discovery of missing actor responsibilities. Domain descriptions are traditionally written in the passive voice. Such formulations omit the subject of the statement. Determining the subject from the context is sometimes possible by performing a specific action. In other cases, such

Table 6.4: Missing coordination acts related to the transaction pattern style

| | Missing definitions | Required Definitions | Percentage of missing information |
|---|---|---|---|
| Standard Transaction Pattern C-Acts | 706 | 1418 | 66.76% |
| Revoke Steps | 93 | 1323 | 93.43% |
| Complete Transaction Pattern C-Acts | 799 | 2741 | 80.10% |

information cannot be determined from the text, and it must be discovered by talking to domain experts or people conducting the processes in the organization. In the Arbitration Court case, of the 38 actor roles participating in the process, six actor roles were subject to this problem. This means that six transactions missed clear requirements with respect to who should perform them. This problem is also been encountered in other procedural law cases. However, as we did not design an experiment to measure these a priori, we are to provide quantified information here and present it as a general observation contributing to our conclusion. Applying the OER method as described in Section 6.1.1 leads the modeler to uncover cases of missing actor responsibilities and allows the specification of user roles for a target software system, thereby increasing the quality of the software requirements.

The second observation was that some roles are too generic, and which role within the organization should fulfill it is unclear. As explained in Section 2.3.2.1, in the DEMO models, the actor roles are formulated in a generic manner, as a "unit of authority" participating in the transaction. Therefore, discovering specific actors in transactions was not part of the experiment; however, based on the discussions with the modelers involved in the experiment, the textual process descriptions were determined to be, in many cases, missing specifications of these actors or provided only generic specifications ("court"). Unclear actor roles can cause problems for a party that wants to request a transaction and must know to whom they should make the request. However, this is not considered a significant problem, as this type of detailed information is typically provided by other documents within an organization (procedure protocols).

#### 6.4.2.3 Complexity Management

Essential high-level DEMO models may reduce the complexity and improve the readability of the process while maintaining the desirable properties of formal languages, as explained in Section 2.3.3.3. The reduction in complexity can be observed in the arbitration court case study presented in the previous section. Figure 6.2 shows a high-level overview of a 40-page process. In other approaches, each transaction symbol is represented by up to 19 activities, because it encapsulates a complete transaction pattern (Figure 2.4). Moreover, this encapsulation improves the possibility of modularizing the resulting models. Figure 6.3 demonstrates this, where Transaction 5 (Hearings Preparation) was modeled as

a subprocess of the parent Arbitral Proceedings from Figure 6.2. Moreover, without this complexity management, in the case of the full scope of the Complete Transaction Pattern, we would be dealing 703 elements, which would be difficult to modularize without clear methodological guidance. For example, this is a typical problem in BPMN, because no such self-sustained building blocks exist comparable to the notion of DEMO transactions, only limited decomposition techniques of activities.

### 6.4.3 Experiment Conclusion

Based on the observations and their implications discussed, we can draw a conclusion regarding the research question formulated in Section 6.1: **Can EE theories be used to increase the quality of business process requirements for process-based information systems?** The answer suggested by our non-trivial experiment is: Yes, EE theories and DEMO methodology can increase the quality of process-based software requirements in the points presented, while considering the limitations discussed in Section 6.1 and the following section.

### 6.4.4 Approach Limitations

While the experiment encourages the application of the proposed method in improving the quality of business process requirements for process-based information systems, some inherent limitations must be considered.

The DEMO methodology is inherently limited in its conceptualization possibilities. It is only applicable for ontological clarification (the procedural aspects of the law), assigning responsibilities, and modeling objects with their properties (the FM). It does not help with any implementation-specific or technology-specific concerns, as they are beyond its scope. However, complementing it with other methodologies and notations is possible, such as in [24];, Cordeiro et al. proposed using UML in conjunction with DEMO. This may further help in revealing the additional ambiguity and lack of clarity hidden in other parts of the domain descriptions.

Next, applying the DEMO methodology requires nontrivial knowledge and practice, which was demonstrated in our experiment. Performing this manually requires considerable elaboration. This can be addressed by applying natural language processing (NLP), as discussed in Section 6.4.6.

Although we abstract this work from most technical topics related to software implementation, we argue that higher-quality software requirements will positively impact the quality of the resulting software system.

### 6.4.5 Comparison with Other Modeling Techniques

The main strength of DEMO is the method-driven discovery of missing information. To the best of our knowledge, no other technique provides a method similar to that of OER; however, methods with similar goals exist, such as the process ontology-based approach

(POBA) introduced in [47]. It begins with the textual description of a process, generates a semantic model of the text, performs an ontology-based validation, and proceeds towards constructing a BPMN process model. The POBA approach was shown to be superior to modeling a process using the BPMN with best practices only.

We can assume that applying POBA on our case studies would be infeasible, resulting in overly complex and intellectually unmanageable models. This assumption is made by considering the size of the ontological model presented in Fig. 5 of [47], which contains 28 elements for a one-paragraph-long process description. Extrapolating from one of the presented cases, 40 pages of the arbitration court case would result in approximately 5 000 elements. Therefore, the resulting BPMN model would be difficult to comprehend, and POBA does not offer any complexity-management method. Conversely, in our approach based on DEMO, by applying the distinction axiom, composition axiom, and a transaction pattern language that addresses the complexity, a significant complexity reduction is achieved, as explained in Section 2.3.3.3, and can be observed in the arbitration court case that can be presented in a one-page high-level model, as presented in Figure 6.2. In addition, the POBA method acts only on information available in a specification text without predefined structural blocks with which to match; thus, it does not offer a clear method for finding missing process steps, unlike DEMO. Thus, POBA presents an interesting ontologically based approach that may be interesting to explore further and attempt to determine solutions to the presented deficiencies for real-life case studies to determine whether the two approaches can be combined.

### 6.4.6 Further Research

The following areas are interesting for further research.

1. Further experiments – Particularly in other domains, conducting more experiments may reveal sufficient textual descriptions available at a necessary level of detail. In addition, quantifying the missing actor roles specification in the current experiment would be desirable, as mentioned in Section 6.4.2.2.

2. Process version management – Software specification are typically followed by change requests. Attempting to extend the approach with methods for managing evolvability may be interesting, such as the Normalized Systems theory. An approach to document management based on this theory is presented in [117].

3. Automation – Conducting the proposed method manually is elaborate. Although ontological analysis cannot be fully automated inherently, the promising research results in the area of NLP could help, such as [26, 47, 173]. Applying such an approach to the first phase of the proposed OER method may reduce the need for manual processing.

4. DEMO models execution – As mentioned, our experiment included the implementation of the process in Camunda using BPMN, and we explained that transformation

from DEMO into BPMN is challenging. It seems that an alternative way of implementation may exist: a direct DEMO model execution. This should be theoretically possible, as DEMO model semantics are well-defined, and could therefore be used to directly generate a workflow system [A.3]. The ontological foundations for software runtime requirements have already been explored in [37]; however, a considerable amount of work remains to be conducted to fill all implementation gaps.

## 6.5 Chapter Summary

Section 6.1 discussed the experiment designed in which the described method was applied to 32 case studies from the procedural law domain. The experiment was conducted over three years, required approximately 2 440 h, and analyzed 115 276 words of legal text. To ensure the high quality of the published dataset, only 23 of the 32 studies (68.75%) were included. One case study, an arbitration court dispute settlement procedure, was summarized in Section 6.2 as a reference example. All 23 included case studies were published on GitHub [96, 144, 143].

The case studies are evaluated in Section 6.4. We determined that applying the method described in Section 6.1.1 is feasible for the creation of formal models with C4E quality criteria (Section 2.3.4) in the procedural law domain. Furthermore, three observations were made that could lead to an increased quality of process-based software requirements. First, the case studies exhibited an average of 80.10 % missing coordination acts. Second, missing actor roles were identified. This was quantified only in the arbitration court case, where six of the 38 actor roles were missing. Third, the properties of high-level DEMO models explained in Section 2.3.3.3 may reduce the complexity and improve the readability of the process models.

Based on the evaluation of the case studies, applying EE theories and DEMO methodology was concluded to increase the quality of process-based software requirements. Section 6.4.4 presented the limitations of the conclusion. The proposed method does not help with any implementation- or technology-specific concerns, as they are out of scope. Another limitation is that applying the method requires nontrivial knowledge of DEMO methodology and practice. Based on these observations, Section 6.4.6 suggested future research goals.

Compared with similar approaches (Section 6.4.5), our experiment provides a large dataset published on this topic with 115 276 words. Other known methods showed only smaller examples with up to 300 words. The proposed method provides a clear method for determining missing process steps and actor roles, while offering strong guidelines for complexity management. This was not observed in similar approaches.

# Part V

# Decentralized Compliance Management

Chapter 7

# Exploring a Role of Blockchain Smart Contracts in Enterprise Engineering

Blockchain (BC) is a technology that introduces decentralized, replicated, autonomous and secure databases. A smart contract (SC) is a transaction embedded into a blockchain containing executable code and internal storage, offering immutable execution and record keeping. Enterprise Engineering (EE) examines all aspects of organizations, from business processes, informational and technical resources to organizational structure.

The blockchain is mainly known as the underlying technology of Bitcoin, but since its introduction, there has been a wide variety of applications. Due to the solutions, it brings to problems such as the double-spending and Byzantine Generals' Problem, blockchain has been called a breakthrough in the computer science [149]. Blockchain 2.0 enhances the application of blockchain beyond cryptocurrencies and introduces concepts for flexible and programmable transactions referred to as smart contracts. Smart contracts enable the creation of more complex decentralized applications (Dapps) and even decentralized autonomous organizations (DAOs) on the blockchain.

The automation of SC creation could be a great benefit, as it would bring a level of security. As explained in the paper by Alex Norta [136] referencing a crowdfunding project that was hacked because it contained security flaws, resulting in a $50 million loss. "The incident shows it is not enough to merely equip the protocol layer on top of a blockchain with a Turing-complete language such as Solidity to realize secure smart-contract management. Instead, we propose in this keynote paper that it is crucial to address a gap for secure smart-contract management pertaining to the currently ignored application-layer development." [136]

Therefore, the blockchain and smart contracts have been subject of interest concerning the discipline of Enterprise Engineering (EE) and the usage of smart contracts in the DEMO methodology, enhancing the creation of Dapps.

This chapter is organized as follows: In Section 7.1, compatibility of BC and EE is evaluated. In Section 7.2, principles to devise smart contracts from DEMO models is proposed. A proof-of-concept case is provided in Section 7.3. The related research is in

Section 7.4. In Section 7.5, the current results are summarized.

This chapter was published and presented at the EEWC 2018 conference in Luxemburg [A.4] and is part of the DSR cycle 2.1.

## 7.1 Evaluation of BC and EE Compatibility

The first important thing to realize is that Enterprise Engineering [31] is a scientific discipline with an underlying enterprise modeling methodology for transaction modeling and analyzing and representing business processes (DEMO) [29]. On the other hand, blockchain and smart contracts are a technology. However, from the nature of the problems they are both addressing and even from the underlying terminology they use, it seems like they could be used together. This is a more challenging question, and a thorough understanding of both of them is required to bring about a correct way of using them together.

### 7.1.1 Smart Contract Misconceptions

#### 7.1.1.1 Autonomous Smart Contracts

The idea that smart contracts can operate fully autonomously is partly true, but more in the sense of immutably following a stated logic rather than performing actions independently. Smart contracts are not programs that are active all the time, they are pieces of code that are run only when invoked. In Ethereum, this is possible either by sending a transaction or a message to the contract's address. So, the idea often presented, that a smart contract actively waits for some event (a certain date) and then executes itself is a misconception [130].

#### 7.1.1.2 External services

One of the prevalent attributes we find when researching smart contracts is that they are designed to use external data. However, this is not that easy to achieve and it is given by the very principle of determinism, which is an essential feature of a blockchain. When running a smart contract, all nodes must come to the same result, therefore they must operate on the same data. Using external data sources to gather data for a smart contract's execution is impossible, as we cannot be sure that the same data will be served to all nodes. Secondly, smart contracts cannot be self-initiated.

All data used must be determined at the invocation of a smart contract. Data must be sent to the contract as a parameter of the invoked function, or produced by so-called *oracles* [12]. Oracles are an Ethereum design pattern, and they serve as "the interface between contracts and the outside. Technically, they are just contracts, and as such their state can be updated by sending them transactions. In practice, instead of querying an external service, a contract queries an oracle; and when the external service needs to update its data, it sends a suitable transaction to the oracle." [11] This is a standard solution to

the need for external data, but the fallback is that we again rely on a centralized external service that we need to trust.

Furthermore, smart contracts should not initiate any action outside the blockchain. For example, it might be an idea for a smart contract to call an external API when some condition has been met. At this moment, there are at least 23880 [44] active nodes in the Ethereum network. All of these independent nodes are executing the same smart contract code, so it would result in 23880 API calls with the same request. Another problem is that the source code of a smart contract is public, and it is running on an untrusted machine, so anyone can fake the API call. That is what smart contracts or any blockchain transaction should not be used for. One must understand it is not an executional system, it is more of a notarization system or controlling system, a trustless database.

#### 7.1.1.3 Privacy issues

As a public blockchain is a distributed database, there is no access control to the data and actions it holds. Every node can see everything: transparency by nature. Therefore, it should be considered thoroughly what to store in a public blockchain and to ensure the security of confidential information.

### 7.1.2 BC as a Transaction Execution System

BC smart contracts can be used to execute the DEMO models because they are represented by a Turing-complete programming language. There are two options for how to do that.

The first option may be to implement the whole DEMO transaction execution on the blockchain through smart contracts. With all the limitations and misconceptions introduced earlier, it might not be possible to implement complete business logic, and there is no need to run the exact same transaction execution multiplied on thousands of computers. Furthermore, transaction execution on a blockchain is not always without expenses, this may vary based on the platform used. But in general, choosing this approach is questionable because we have applications in "regular" programming languages, which once developed, are free of cost.

The second option may be to choose only some transactions, of which full or partial execution on a blockchain would bring benefits. We can make use of blockchain's notarization of SC code and secure a trustless transaction execution when operating with untrusted third parties or multiple organizations. The remaining transactions are executed by a standard EIS. For example, we have a contract that states once a certain amount of money is paid, an asset will be transferred by an external company. This alternative makes use of the primary benefits that BC technology brings.

### 7.1.3 BC as a Notarization System

Based on the definition of the Blockchain: "It introduces decentralized, autonomous, replicated and secure database, that based on cryptography offers trustless network without a

need of intermediary", another application of BC in EE could be to serve as a notarization system. Smart contracts can offer notarization of documents, agreements, and all information related to transactions, progress, and results of transactions. BC could then provide a consistent and reliable source of data, facts, and transaction states for all parties involved in the process.

## 7.2 Principles to Devise SC from DEMO Models

In this section, we discuss the principles of creating BC smart contracts based on the DEMO methodology. Next, we introduce a software architecture of an enterprise information system based on DEMO that communicates with a smart contract.

### 7.2.1 SC based on DEMO

In the previous section, we introduced what could be a possible usage of smart contracts in EE. We explained that there are two possible approaches that can also be combined:

- *Notarization* of documents, agreements, and all information related to transactions, progress and results of transactions.
- *Trustless execution* of transactions or a part of transactions.

As mentioned before, the decision of whether to use SC for the process implementation is individual for every case. In general, a good use case could be to use it when operating with untrusted third parties or multiple organizations. There is no such need for notarization or trustless execution within the internal business processes, but the need arises when dealing with transactions on the border of the scope of interest when communicating with external actors. A BC smart contract can also represent the coordination point between an internal IT system and external actors.

### 7.2.2 DEMO Transaction As Contract

A DEMO transaction is represented as a contract in a blockchain. The contract has its own address, internal storage, attributes, methods, and it is callable by either an external actor or another contract. This is the functionality needed to represent a DEMO transaction. In this paper, we implemented the execution of DEMO transactions according to the DEMO Machine [A.3] and associated theories. For mapping contracts to the corresponding DEMO transaction, we use the names defined in the Transaction Product Table. The contract then encapsulates the transaction notarization or execution.

### 7.2.3 Notarization

Notarization of a DEMO transaction facts can be divided into two parts:

- Notarization of the transaction P-facts and documents.
- Notarization of the transaction execution C-acts and C-facts.

In the first case, we are looking at using a smart contract as storage of facts. To construct a smart contract carrying transaction facts, we can combine information from three models: Organization Construction Diagram, Bank Contents Table, Object Fact Diagram, and Action Model. From them, we can retrieve which object facts are needed for the transaction, and wherein the transaction execution arise, we can evaluate the changes of the objects associated with the transaction execution. An object class can be represented as an internal state variable in the contract with the corresponding name from the DEMO model. A smart contract then serves as a storage of facts (database) for the transaction.

In the second case, we want to notarize the transaction execution. We use the complete transaction pattern from where we take all possible C-Facts and add their representation into the contract. The contract then holds its current state as a C-fact. For every C-Act, we create a contract method that changes the contract state to the corresponding C-Fact. Every change of a C-Fact issues an Ethereum system-wide notification (Event), allowing external systems to keep track of their contracts. Once the transaction is completed and the P-fact is created, another event is emitted stating the P-fact.

### 7.2.4 Transaction Execution in SC

To implement a transaction execution in the contract, we need to understand the whole DEMO model and the relationships between transactions.

In the PSD model, the response links and wait links are specified, and we can see enclosed transactions. Action rules are guidelines for dealing with the events that actors have to respond to. In practice, they are referred to as business rules [33], so they can be used to construct the execution logic. If we execute a certain C-act in the contract, we look at the action rules that contain this C-act, and we construct the corresponding method accordingly. Depending on the actor roles we define the executor of the method, we define general conditions based on the transaction pattern, such as that to perform a promise, the contract must be requested. Finally, we translate the action rule pseudo-code to a contract code. In this case, we also need to add the notarization of the fact on which the transaction operates. Method to execute a C-act is named after the C-act concatenated with the transaction name, e.g. `promiseMortgageCompletion`.

If a transaction encloses other transactions, we have to decide how to handle them. There are three possible solutions. Firstly, the sub-transaction can be implemented as another contract. The enclosing contract then stores the address of the sub-contract. This way, when the action rule contains a response link for the child transaction C-act, we call the corresponding C-act method of the child contract. The sub-transaction can also store a reference to the enclosing transaction to implement the wait links. Secondly, there must not always be a need to create a separate contract for sub-transactions, we can implement the sub-transaction inside the main contract. This can be convenient if we are interested

only in a partial execution on BC for the sub-transactions. Finally, the last option is that we do not handle the sub-transactions at all and leave this outside of BC.

### 7.2.5 Extending the DEMO model

Using SC with DEMO is part of the implementation of the organization. From this point of view, the essential DEMO models should not change when using SC. SCs only implement transactions, and in some cases, an actor role can be assigned to it. In both cases, the underlying essential DEMO models are not affected.

As the implementation of a contract can be derived from the DEMO models, the creation of the contract code could be produced from the DEMO models. In this case, we would define a way to identify the transactions and types of their integration on the BC. A solution could be to introduce Transaction Blockchain Table (Table 7.1) that maps transactions to their BC implementation. As for the actor roles assigned to SC, this would be defined in the Actor Function Matrix.

Table 7.1: The transaction blockchain table template

| **Transaction** | **Fact notarization** | **Transaction notarization** | **Execution** |
|---|---|---|---|
| Transaction kind | List of facts to notarize | Yes/No | List of C-Acts to execute |

### 7.2.6 Software Architecture

An implementation of an IT system consists of two parts:

- Enterprise information system (EIS): An IT system that supports the business processed modeled in DEMO. It executes the transactions that do not need to be available in BC.
- Blockchain: A blockchain that executes desired DEMO transactions. It also serves as an audit log to keep the history of desired C-Facts.

The architecture of an IT system integrating the enterprise information system and blockchain is illustrated in Figure 7.1. The EIS contains mainly a business process management (BPM) engine and a blockchain API. The BMP engine is a transaction execution system. The blockchain API is an interface for communication between the BPM engine and the blockchain. The communication with blockchain is carried out through a blockchain node. It contains a transaction processor and a blockchain database, which holds the smart contracts or blockchain logs. The DEMO models and methodology also serve as an orchestrator for the cooperation of the components (BPM engine, blockchain

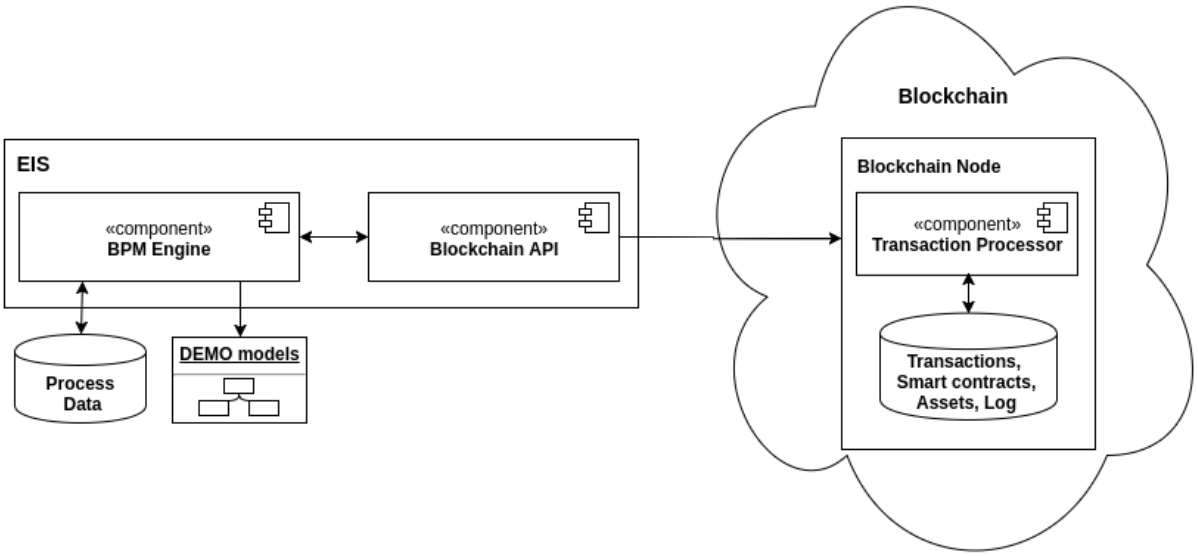


Figure 7.1: Architecture of an IT system based on EE and BC

API and smart contract). They all use the transaction patterns, transaction names, facts, etc., as defined by DEMO.

### 7.2.7 EIS and BC communication

The communication between the EIS and SC is a one-way interaction based on the principles described in Section 7.1. As BC cannot return values or call external services directly, all the interaction is handled from the EIS side.

The EIS contains an API for communicating with blockchain, such as web3.js. This API facilitates the contract deployment, sending of transactions to the contract, getting data from the contract. Using the events mechanism, the API monitors the blockchain log, and "listens" for certain events. This way, the API can watch the change of transaction state or results of contract execution and act on it, mostly if it is a transaction involving external actors.

## 7.3 Proof of Concept

In this part, we describe a proof of concept using a financial transaction, the process of a mortgage, implemented in the Ethereum Solidity programming language for smart contracts. Due to space limitations, we only present the conclusions. The whole proof-of-concept description, as well as a smart contract source code, can be found in a GitHub

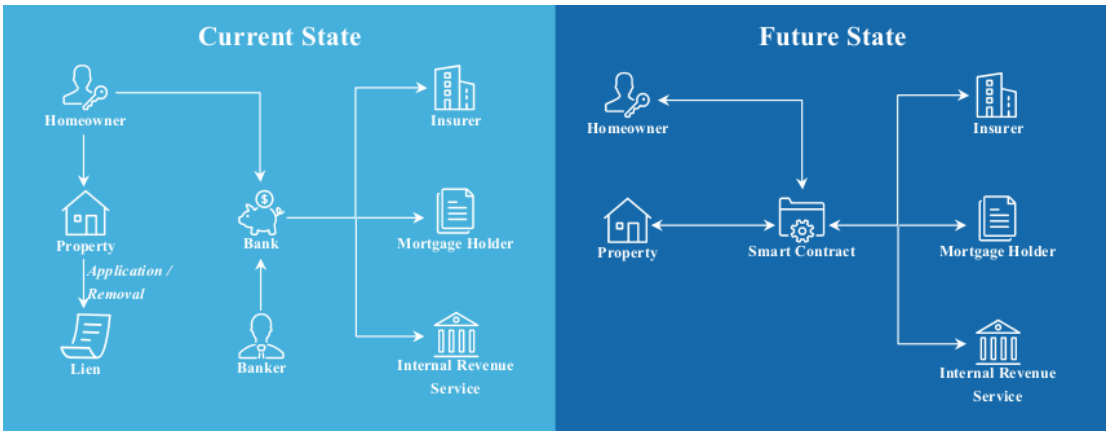


Figure 7.2: Mortgage process changed using smart contract [146]

repository[1]. The example is based on a presumption that a property ownership record is held in a public blockchain. This is a relevant case for the developing countries where it can bring many benefits as described in [119].

The mortgage contract is a complex process involving several parties, dependent processes, level of trust between parties and a lot of documents proving results of auxiliary processes; Notarization is involved for all parts. These aspects all contribute to the overall complexity and costs of the process. Thus it appears as a good use case, where modeling by DEMO would capture the essence of the process and a smart contract could offer an automated notarization, data sharing between parties, and payment processing, thus reducing the need for manual processes, as illustrated in Figure 7.2.

The description of the mortgage process is based on a review of several online mortgage guides [69, 101] and consultation with a real estate agent. The description of the process was then modeled with DEMO to fully understand and illustrate the process. Here we omit the typical pre-approval phase.

We have used the DEMO methodology to understand and reveal the essence of a mortgage process. Using the proposed principles of creating a contract introduced in Section 7.2, we were able to create working smart contracts based on the DEMO models.

We created a trustless notarization of the process. The contract ensures the immutability of the agreed mortgage conditions, such as the amount of payment and interest rate. Further, it controls the execution of some parts, such as automatic mortgage payment control and automatic lien release request. This way, the Client can be sure that once the mortgage has been paid off, the lien will be released. It also defines a single point of access to the data and coordination for all parties, as well as simplifies some steps, as automatic control can be performed, thus allowing us to carry out some acts tacitly. Using smart contracts does not change the essential DEMO model, as they belong to the implementation, which becomes simplified. For example, the Client would not have to bring the confirmation about insurance to the Loaner, because this is done by the smart contract

[1]https://github.com/MIDNP/DemoBlockchain

– this reduces the overall process steps behind the mortgage and eliminates bureaucracy, lags, and errors.

The smart contract implementation uses only a standard transaction pattern. An extension to the complete pattern would be analogous. The pattern is implemented as a state machine, and the model constraints are implemented by function constraints and require() function.

## 7.4 Related Research

#### 7.4.0.1 SC based on BPMN

BPMN is one of the most widely used standard modeling notations and there have been efforts to use BPMN for smart contract implementation. One of them is described in a paper by Weber et al. [168]. The paper elaborates a similar approach to ours of implementing a business process using BPMN on blockchain. It recognizes two alternatives of using blockchain as "a choreography monitor, it stores the process execution" [168] or "as an active mediator among the participants, it coordinates the collaborative process execution." [168]. The approach then introduces a method of translating a BPMN model into a smart contract. This method is mainly addressing collaborative process execution for participants with lack of trust.

To compare, the approach described in this paper may help to prevent possible errors or unwanted states due to the C4-ness quality criteria [29] of demo models. This is very important since the history of a smart contract can't be changed and the code is hard to update once deployed. But in general, both solutions introduce similar findings and principles about the usage of BC and process modeling and a method of translating the models to smart contracts. In the end, it comes down to the comparison of DEMO and BPMN [A.2] itself and evaluating the appropriateness of their use and ability to cover all possible situations when modeling processes.

#### 7.4.0.2 SC based on Petri nets

Another interesting solution can be found in the work of García-Bañuelos [19]. This paper focuses on an optimized execution on the blockchain. It defines a method of transformation of BPMN processes (modeled in a subset of the BPMN standard) into smart contracts through the use of optimized Petri nets. "The method takes as input a BPMN process model. The model is first translated into a Petri net. An analysis algorithm is applied to determine, where applicable, the guards that constrain the execution of each task. Next, reduction rules are applied to the Petri net to eliminate invisible transitions and spurious places. The transitions in the reduced net are annotated with the guards gathered by the previous analysis. Finally, the reduced net is compiled into Solidity." [19]. The work focuses on encoding the control-flow and evaluation of data conditions, however it does not discuss how participants would be bound to the contract instance, and access control.

## 7.5 Chapter Summary

This chapter discusses the possibilities of applying the EE theories and DEMO methodology together with blockchain technology. A proof of concept blockchain smart contract was developed based on a mortgage process. We have defined and analyzed the process, applied the DEMO methodology, and consequently developed an Ethereum Solidity contract using our principles, showing the possible approach to the topic.

Chapter 8

# Systems Supporting Decentralized Compliance Management

For thousands of years, contracts between people were conducted on paper or other material media and enforced by authorities. If necessary, this approach requires all participants to believe in a central authority that enforces the contract obligation. However, today, an enforcement process can take years to settle and cost a significant amount of money in administration and attorney fees.

Recent developments in blockchain (BC) technology have allowed the creation of contracts between people specified in a software code called Smart contract (SC). An SC is enforced by a network of computers that is guaranteed to execute the code adequately based on proven cryptographic algorithms. This technology was a breakthrough in computer science; however, it seems that it has not delivered its full potential. Nevertheless, the most famous SC platform Ethereum [43], is still in the experimental phase and is not ready for mass-scale adoption.

There are numerous challenges to the mass adoption of SC technology. This chapter addresses one of them – a software code does not seem to be the best way to specify the contract between people for two reasons. First, non-technical people do not comprehend software code, and therefore they need to trust someone that will write the contracts for them. This is similar to ancient times when people could not read and write and thus had to trust the experts of this special knowledge. Second, the semantic level of software code is too low, and it is challenging to make a high-level comprehension and reasoning – there was already a case of a programming mistake in the SC, which resulted in a loss of $50 million [111]. We argue that these issues can be addressed by applying modeling methods coming from the discipline of Enterprise Engineering (EE) [30].

A visual language for modeling smart contracts, DasContract, is proposed in this chapter to address these challenges. It argues that by using a visual domain-specific language (DSL), the readability of smart contracts would be more comfortable, and using the model-driven engineering (MDE) approach generates the smart contracts' source code. The proposed approach is demonstrated in two complex case studies where the contract

is designed, generated, and simulated in a blockchain environment. First, a decentralized mortgage process is introduced where banks are made obsolete. A second case study is a decentralized version of EU elections.

The chapter is organized as follows: In Section 8.1, an approach for blochchain-based contracts between people is introduced. A visual domain-specific language DasContract is presented in Section 8.2. The proposed approach is demonstrated on a Mortgage case study in Section 8.5 and EU elections case study in Section 8.6. Limitations of our approach are discussed in Section 8.7. The related research is discussed in Section 8.8. Finally, the current results are summarized in Section 8.9.

All the work described in this chapter is available as an open-source project DasContract [139]. Parts of the work were published in [A.5, A.7, A.6] and contributions were made in supervised theses [A.13, A.15, A.29, A.17, A.30, A.16, A.28, A.27]. The presented work is part of the main decentralized DSR cycle and the DSR sub-cycle 2.2.

## 8.1 Overview of Our Approach

The contracts are defined as *"An agreement between two or more parties creating obligations that are enforceable or otherwise recognizable at law."* [53]. The main goal of this section is to argue that the contracts can be made better than using only plain text. Especially the contracts in the modern digital world wherein most countries the same law still applies for the online and offline world. By saying "can be done better," we mean that a large amount of the repetitive administrative work can be eliminated, the comprehension can be increased, and third-party involvement reduced (courts, distrainors, ...).

To limit our scope of interest, we will attempt to bridge the legal and technological worlds by providing a formal language to define smart contracts.

### 8.1.1 Contract Maturity Model

To measure the contracts' quality, we propose a contract maturity model. This model is focused on capturing the accuracy of a mutual understanding in contract representation. A framework for a language evaluation proposed by Giancarlo Guizzardi [60] is used as a basis of this model. The Figure 8.1 shows the relation between conceptualization, abstraction, modeling language and model [60]. In law, the most used modeling language is plain text. Plain text has great expressivity, but it can lead to ambiguity and undermine clarity.

**A Verbal Contract** is the oldest form of contract between people. The terms of contracts are agreed upon in a natural language that both parties understand and store in their brains. This form is excellent for a small number of people with shared domain conceptualizations who trust each other.

**A Written Informal Contract** is a version of a verbal contract that is written on a persistent medium in the form of a natural language. This simple act ensured that there was only one possible contract model. Sadly, each participant can interpret the natural text

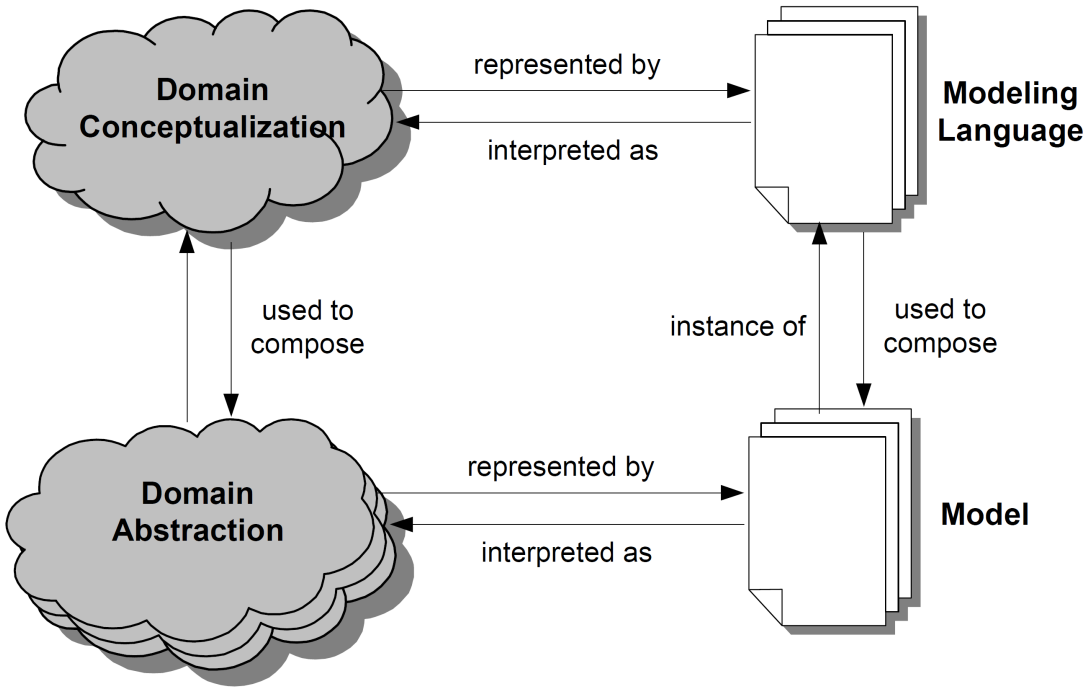


Figure 8.1: Relation between conceptualization, abstraction, modeling language and model [60]

differently because they may have different domain abstractions. In this type of contract, it is still possible to create illegal contracts because no legal framework is followed.

**A Legally Binding Contract** is a written contract that follows a legal framework. In this case, the legal framework and a natural language act together as a modeling language to create a model. However, the law framework is usually defined in a natural language, and it may be hard to create a model due to multiple domain conceptualizations. This ambiguity makes it harder to compose a model and validate its compliance with a modeling language.

**An Ontological Contract** is a form of contract that controls the domain conceptualization and abstraction. The modeling language is represented by a domain conceptualization which allows only one possible interpretation. A composed model based on such modeling language has a clearly defined domain abstraction, and therefore the ambiguity can be controlled. This means that all involved parties work with a shared domain conceptualization and abstraction because it is stated explicitly.

### 8.1.2 The Concept Architecture

This chapter builds on top of the work presented in Chapter 7 where possible use of DEMO methodology to model blockchain smart contracts was explored and demonstrated on a mortgage case study. This chapter narrows the use-case to modeling legally binding

| Maturity | Name | Contract Form | Accuracy |
|---|---|---|---|
| 1 | Verbal contract | A mutual understanding | No written record of a contract |
| 2 | Written informal contract | Informal text | Typically ambiguous interpretation, possible errors, no legal framework |
| 3 | Legally binding contract | Legal text | Risks of ambiguous interpretation, possible errors, legal framework contains ambiguities itself |
| 4 | Ontological contract | Ontological model | Ambiguity effectively controlled |

Figure 8.2: A contract maturity model

contracts between two or more parties.

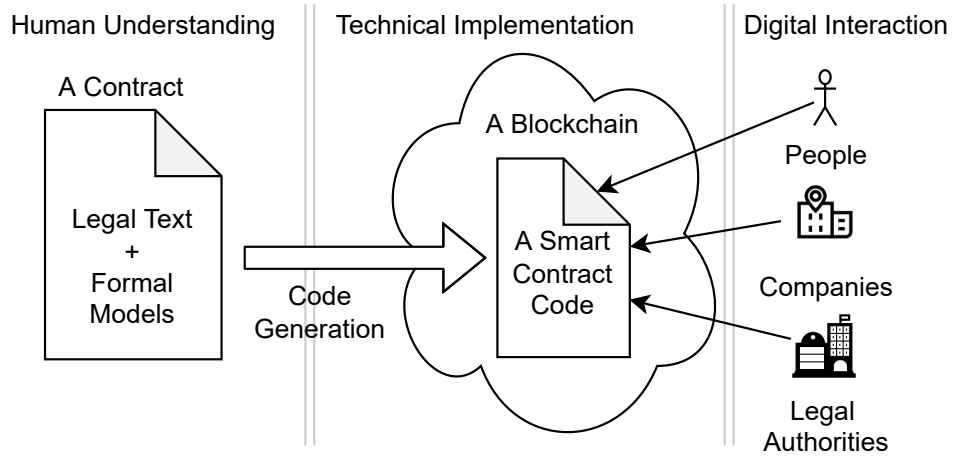


Figure 8.3: A proposed concept architecture

Our proposed approach is described in Figure 8.3. It consists of three parts, in *Human Understanding* the contract is formally and legally specified. The *Technical Implementation* part shows an execution of the contract in the Blockchain. In the *Digital Interaction* part, two or more parties are interacting with the smart contract through their digital devices.

**Human Understanding** part defines a contract between multiple parties that they need to agree on. Such a contract is a combination of legal text and formal ontological models. The legal text, in some form, specifies the legal validity of the formal model.

The formal models need to be unambiguous, so only one possible interpretation is allowed. These formal models are specified in Section 8.2.

**Technical Implementation** part specifies how formal models from the contract are transformed into a software executable code and uploaded into a blockchain as a smart contract. Guidelines and algorithms on how to generate such code from the formal models are provided in Section 8.3.

**Digital Interaction** is a part where people, companies, and legal authorities can interact with the agreed-upon contracts. Thanks to cryptography, since the contract is in a blockchain, the interaction is fully digital and legally binding. Blockchain by design also provides an audit trail of all actions performed by the parties and ensures that the agreed-upon contract is executed correctly. This part is demonstrated on case studies in Section 8.5 and Section 8.6.

### 8.1.3 The Proposed Method

Our approach was designed to answer the research question 2 from the decentralized DSR cycle. It adopts the BPM methodology to provide a methodical way of designing a blockchain smart contract. A Model-driven engineering approach generates a smart contract source code for a particular blockchain. This thesis uses an Ethereum smart contract platform, but the approach can be generalized. An application of our approach is shown in Section 8.6 on a process of EU parliament elections and a mortgage process case study in Section 8.5.

The BPM life-cycle in the context of digitizing a process to be used in blockchain technology is following:

1. Design - A process is designed in a DasContract language with blockchain limitations in mind. Compared to a traditional software system, the major limitation is its immutability and the requirement that all performed actions are deterministic.

2. Modeling - A simulation of the DasContract may be performed to ensure the correct behavior of the process. This may be critical because the contract cannot be changed after being deployed, and it may be handling significant monetary or legal value.

3. Execution - A smart contract source code is generated and uploaded to the Blockchain. Because the metamodel is implementation-independent, any supported blockchain platform can be chosen.

4. Monitoring - Due to the inherent blockchain capability to record all transaction history, auditing and analyzing process execution history can be made.

5. Optimization - The optimization is not relevant in Blockchain because once the smart contract is uploaded, it is impossible to change it.

6. Re-engineering - Similar to optimization, re-engineering is not available. A new process can be designed and uploaded to the Blockchain, but the old one will still be running.

## 8.2 DasContract – a Visual Smart Contract

In this section, we introduce a system that aims to materialize the proposed concept architecture introduced in Section 8.1.2. The DasContract system consists of three parts. The human understanding part introduces a visual domain-specific language called DasContract that allows people to specify formal models of their desired contracts. Guidelines on how to model the procedural law were introduced in Chapter 6. The second part is about blockchain smart contract generation from the DasContract language based on the MDE principles. Furthermore, the last part is about the digital interaction of people, companies, and legal authorities with the generated blockchain smart contracts. A proof-of-concept implementation of the proposed system is available as an open-source project on GitHub [139]. This section is focused on introducing a specification of *Formal Models* as it was introduced insection 8.1.2.

### 8.2.1 DasContract Model Specifications

The DasContract consists of three interconnected models: process model, data model, and forms model. The metamodel of the data and forms models is shown on Figure 8.6. The highlighted parts in blue are the recent additions to support the representation of fungible and non-fungible tokens and enums. The tokens inherit from the entity, and therefore they can have data properties to represent token metadata.

**Process Model** shown in Figure 8.5 specifies the contract's process activities, their execution order, property mappings, and user roles. The model is based on an extended BPMN 2.0 level 3 notation subset. The major addition is support for blockchain tokens that allow issuing and receiving both fungible and non-fungible tokens. During the transformation, the process sequence is transformed into a smart contract programming language statements such as if-else, functions, etc. This process may vary for different blockchain implementations. The process model also excludes the concepts from the BPMN that cannot be implemented in Blockchain due to its technological capabilities.

**Data Model** shown in Figure 8.6 is a domain model of the process is based on the UML class diagram. It allows specifying entities, properties, and relationships between them. The properties may contain primitive types such as int, bool, and string, but arrays and enumerations are also supported. Address and AddressPayable types are added to support the storage of a blockchain actor's addresses and consequent token or cryptocurrency transfers. A blockchain token is represented as a special type of entity and supports defining both fungible and non-fungible tokens with custom properties. A transformation of the models to the smart contract source code is straightforward because the blockchain programming languages support such concepts in the form of classes, structures, enums, and properties. The support of tokens is native on some platforms. On other platforms such as Ethereum is added as a third-party library.

**Forms Model** shown in Figure 8.6 specifies an interface for the user activity input. The forms model is generated into two parts during the generation – off-chain model and on-chain code similar to a web application client-side and server-side code. A blockchain wallet interprets the off-chain model that allows the user to interact with the smart contract. Most blockchain implementations currently have no support for off-chain code inside a smart contract. The on-chain code handles validations, user rights, and property bindings.

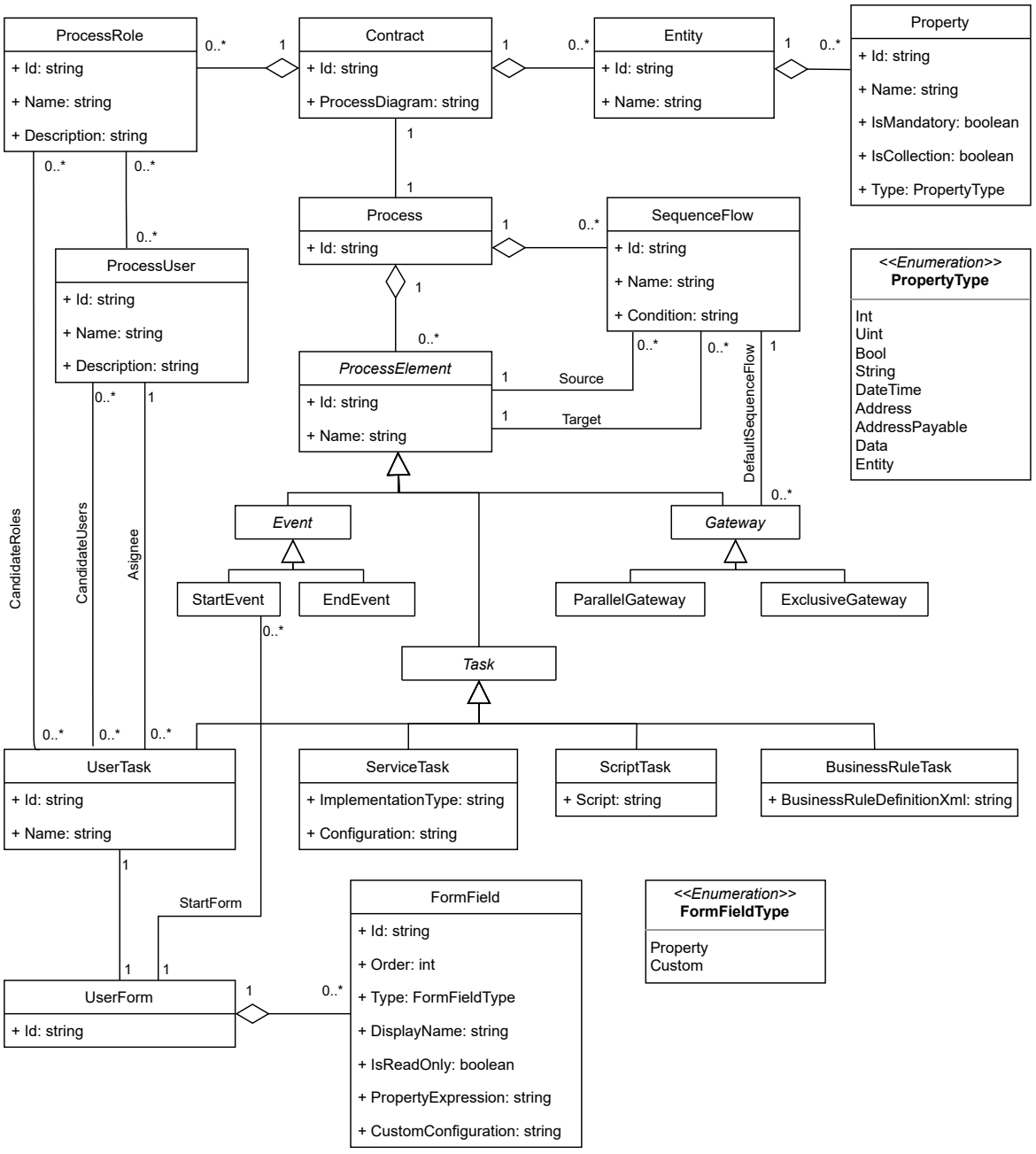


Figure 8.4: A DasContract high-level metamodel

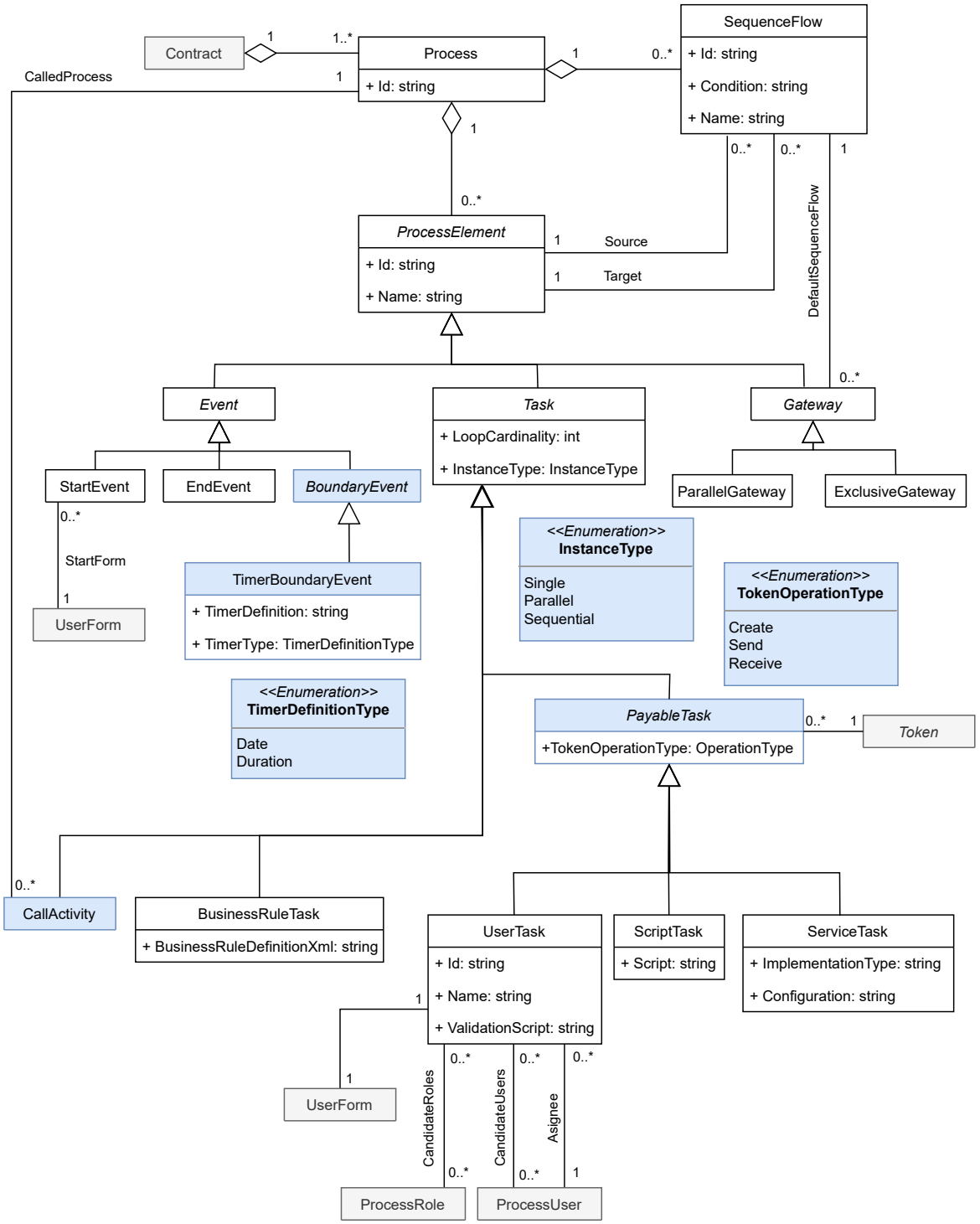


Figure 8.5: A DasContract process metamodel

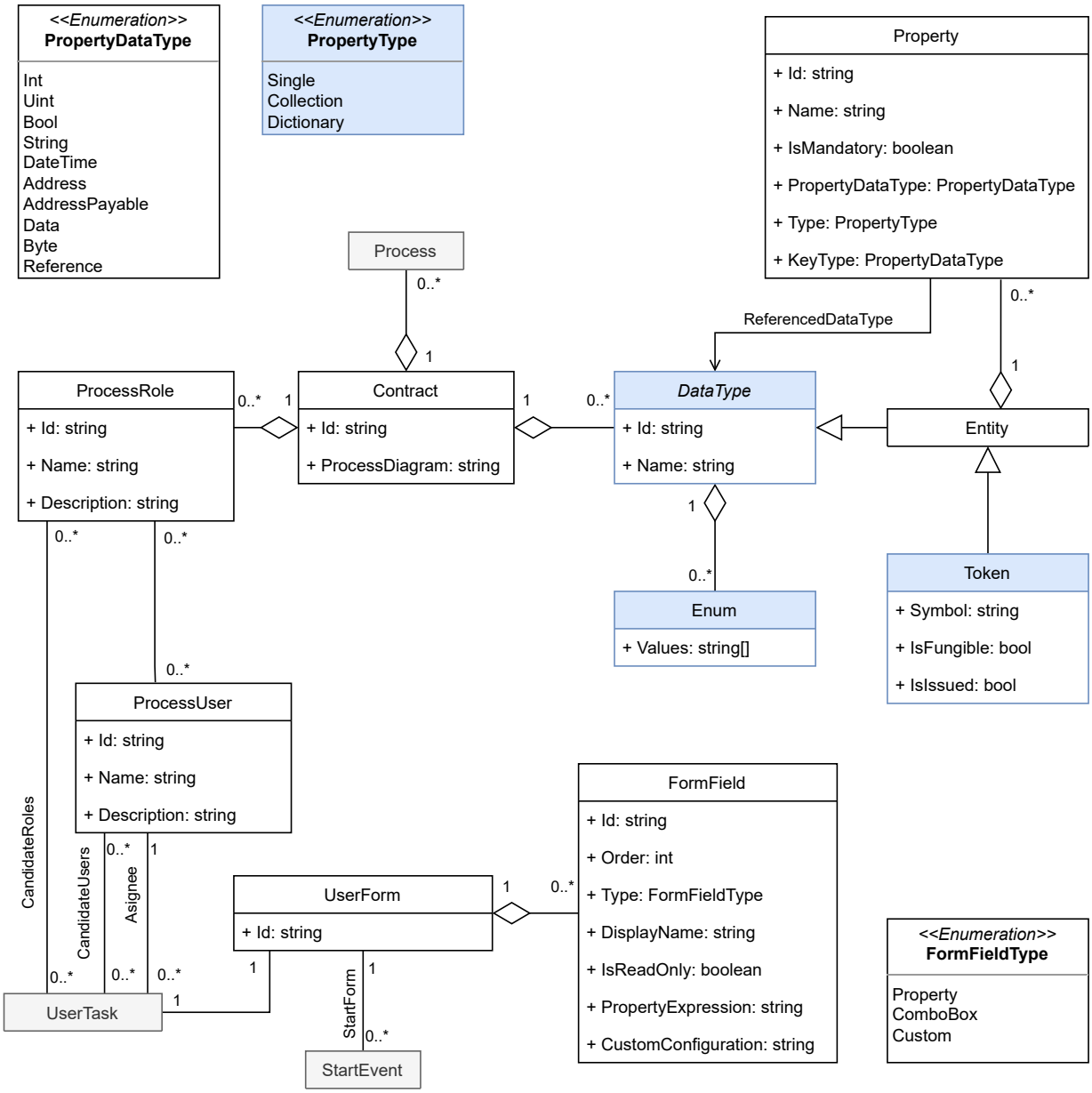


Figure 8.6: A DasContract data and forms metamodel

### 8.2.2 DasContract Model Editor

To design the DasContract models, a visual editor was created as a part of the open-source project [139]. The editor allows stakeholders together with a DasContract expert to design their desired contract and generate blockchain smart contract. Currently, a generation into Ethereum Solidity language is available. However, there is an ongoing development of support for the Plutus language on the Cardano blockchain.

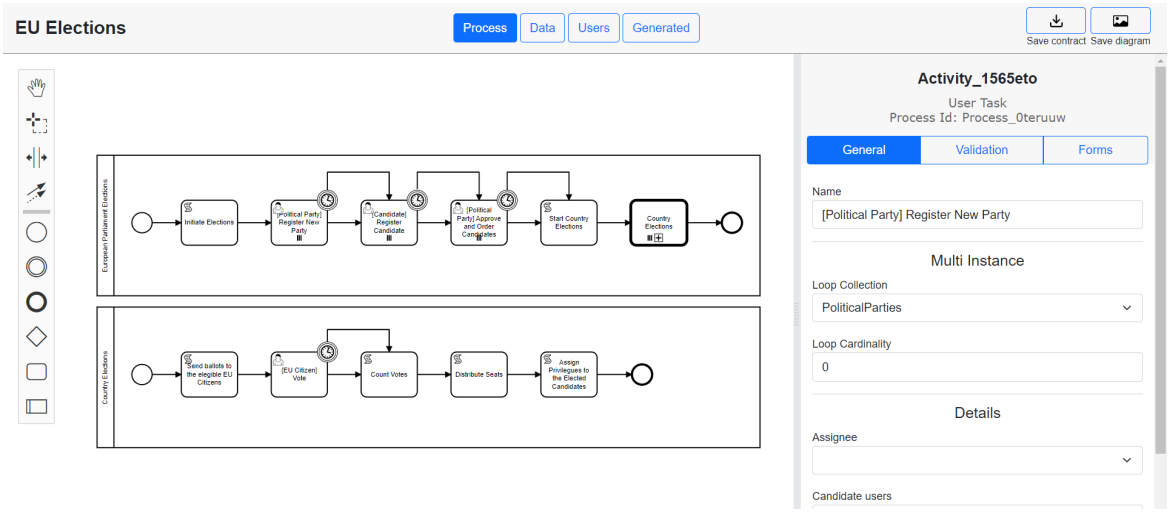


Figure 8.7: A DasContract process model editor

## 8.3 Code Generation and Execution of DasContract Models

This section introduces a way to generate blockchain smart contracts using a Model-Driven Engineering approach (MDE) [10]. To achieve this, a way of automatically generating blockchain smart contracts is described. Ethereum Solidity language is used as a demonstration; however, the principles described should be transferable to other blockchain implementations.

An extended subset of BPMN 2.0 level 3 is used as a basis for the execution behavior. However, compared to the BPM systems that interpret the model, an approach that generates code with the model behavior is used. This approach is used due to blockchain performance and storage limitations. Each line of code executed in the Blockchain costs money, and the storage costs are also at a premium.

The algorithm described in this thesis is implemented in C# programming language and published on Github under an MIT license [139].

The DasContract DSL consists of the following models: *Data Model* Specifies data structures used inside of the smart contract. These structures can be referenced inside of the process model. *Process Model* Specifies a contract's business process as an extended subset of the BPMN 2.0 language. *Forms Model* Specifies a user form required to fill by the user in a process user task. The forms provide a way to interact with smart contracts.

### 8.3.1 Data Model

Implementing the data model is straightforward because the blockchain smart contract languages provide generous native support to specify data structures. The supported concepts are entities, properties, entity properties, and arrays. Each property can also be

marked as mandatory. The data types of properties are Int, UInt, Bool, String, DateTime, Address, AddressPayable, Data, Entity. The types Address and AddressPayable are Blockchain specific and support cryptocurrency or token payments.

**Example** Let us have an entity *Payment* with four properties – two addresses identifying the sender and receiver, a numeric defining the amount sent, and a boolean indicating whether the payment was on schedule. The generated code is shown in Listing 8.1.

```
struct Payment {
        uint256 amount;
        bool onSchedule;
        address sender;
        address receiver;
}
```

Listing 8.1: An example of a generated data structure.

### 8.3.2 Process Model

The process model uses an extended subset of the BPMN 2.0 level 3 notation. The formal specification of a BPMN execution is already well researched, and there are many different formalizations such as [34, 92]. In this thesis, the algorithms are based on Petri-net-based formalization described in [35].

Blockchain smart contracts are more similar to a programmable database rather than a desktop, web, or console application. Due to this fact, not all of the BPMN concepts can be implemented or make sense. Therefore only a limited subset is implemented: user task, script task, XOR gateway, parallel gateway, start event, and end event. The blockchain-specific activities were added: payment task. The payment task is a task attribute that can be added to both user and script tasks to add support to work with cryptocurrency or tokens.

**Process Flow** Exclusive gateways are converted into a sequence of if-else statements, advancing the execution based on the statement's result. Parallel gateways are more complex, as they can not only create multiple execution branches but are also used for synchronizing multiple flows (branches) into a single flow. If a gateway has multiple incoming flows, then a counter is defined, keeping track of the number of flows that have reached the gateway. The outgoing flows are triggered once the counter matches the number of incoming flows. An example can be seen in Listing 8.2, which contains the logic of parallel gateways generated based on Figure 8.8.

```
int Gateway_2Incoming = 0;

function Gateway_1Logic() internal {
        ActiveStates["EscrowPropertyRights"] = true;
        ActiveStates["AcceptInsurance"] = true;
        ActiveStates["EscrowMoneyPayable"] = true;
}

function Gateway_2Logic() internal {
        if(Gateway_2Incoming==3){
                ActiveStates["ValidateContractPayable"] = true;
                ValidateContractPayable();
                Gateway_2Incoming = 0;
        }
}
```

Listing 8.2: Gateway logic generated based on the model snippet in Figure 8.8.

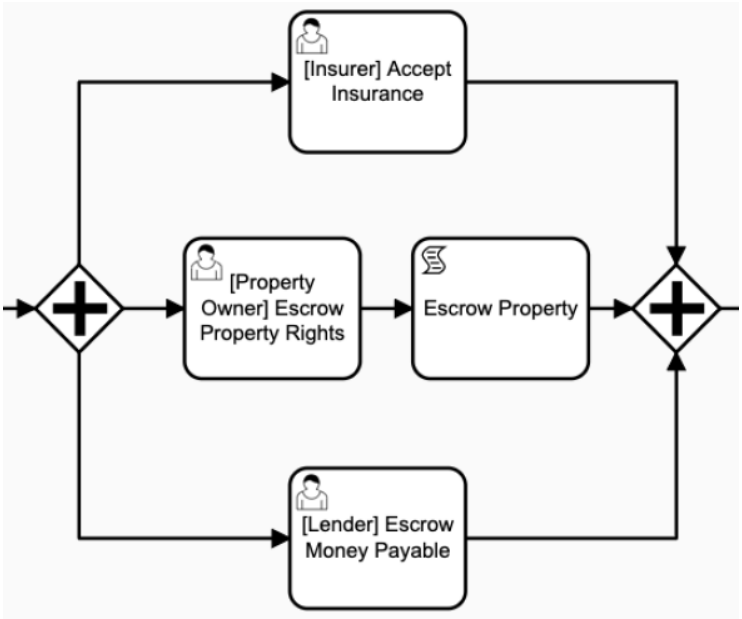


Figure 8.8: A contract snippet used to demonstrate a gateway conversion in Listing 8.2 and user task conversion in Listing 8.3

**Activities** Functions are also used to encapsulate the logic of activities. Unlike the internal gateway functions, these functions are publicly visible. User activities allow accepting parameters and storing them inside the contract using property binding logic. The gener-

```
modifier isEscrowPropertyRightsAuthorized{
        if(addressMapping["Property Owner"] == address(0x0)){
                addressMapping["Property Owner"] = msg.sender;
        }
        require(msg.sender==addressMapping["Property Owner"]);
        _;
}

modifier isEscrowPropertyRightsState{
        require(isStateActive("EscrowPropertyRights")==true);
        _;
}

function EscrowPropertyRights(bool Agree) isEscrowPropertyRightsState
    isEscrowPropertyRightsAuthorized public {
        ActiveStates["EscrowPropertyRights"] = false;
        escrowagreement.agreeToEscrow = Agree;
        ActiveStates["EscrowProperty"] = true;
        EscrowProperty();
}
```

Listing 8.3: User task logic generated based on the model snippet in Figure 8.8.

ator also allows restricting access to function execution based on the executor's address. This is done using roles defined using square brackets inside the name of the activity (see Figure 8.8). The addresses to each role are assigned at runtime, meaning that anyone can execute an activity with an unassigned role. The first execution assigns the role, reserving the remaining activities of the given role to that address.

An example of a converted user activity can be seen in Listing 8.3, generated based on the *Escrow Property Rights* activity in Figure 8.8. The generated function contains two modifiers, checking whether the contract is in the valid state and whether the executor (*Property Owner* role) is authorized. The function has one input parameter that has been defined using the editor. According to the defined property binding logic, this parameter is stored inside the contract. An address of the property owner is read from the *msg.sender* property that provides a sender's public address verified by his private key while sending the transaction to the Blockchain.

**Token Activities** Tokens play a significant role in the smart contract ecosystem by allowing express ownership of an asset. These tokens can also interact with other smart contracts by complying with the token standards. An activity that would allow us to create/send/receive tokens would significantly improve the generated smart contracts capabilities. For example, it would allow sending voting ballots (in the form of a token) to the voters.

### 8.3.3 Forms Model

The forms model defines a form shown to a user and allows interaction with a contract. Such logic is implemented in two places called on-chain and off-chain.

**On-chain code** is run inside the Blockchain. Our algorithm is represented as parameters passed into a function generated from a user task. An example is shown on Listing 8.3 - the method *EscrowPropertyRights* contains a parameter Agree that will be provided by the user while calling the method.

**Off-chain code** is run inside a cryptocurrency wallet to provide the user with a comfortable user experience while interacting with a smart contract. An extended forms model that is used to generate the off-chain behavior is further explained in Section 8.4.

### 8.3.4 Design, Compilation and Execution

A new visual modeling environment was created to model the new DasContract models. The visual modeling reduces the modeler's errors and reduces the time required to produce a model. The modeling environment is available on Github under an MIT license [139].

Compilation to the Solidity code assumes a valid DasContract model created in the modeling environment and does not check for other errors. The last check is done during the Solidity code compilation (usually in the Remix environment) and allows the modeler to fix issues in custom script tasks and user task validation logic.

## 8.4 Extended Forms Model for Digital Interaction

So far, we only explored the on-chain part of the DasContract system that would be equivalent to a backend in traditional web applications. The frontend of the blockchain systems is called off-chain and can be implemented in various forms. Because the blockchain smart contracts will be relatively simple, we explored how to include a model of the off-chain user interface and its behavior in [A.29]. Many approaches were considered, such as techniques used in low-code platforms and various state-of-the-art approaches to SPA web applications such as MVVM.

Based on the analysis, the DasContract forms model was extended with standard UI elements such as combobox, textfield, datefield, and it is shown in Figure 8.9. The fields are connected to the DasContract data model using the *ParamBind* property. A form is connected to the DasContract process model through a user task. During the execution of a generated smart contract code, the information from the extended forms model is parsed on the UI and displayed to the user. The user fills out the form and submits the data to the Blockchain. The generated smart contract logic also validates the user rights to submit the form. An example is provided in the Mortgage case study discussed in the next session in Figure 8.12.

The DasContract editor was extended to support modeling the off-chain forms. An example of the working application is shown in Figure 8.10.

This section contains only a very brief introduction to the off-chain interaction that is possible with the DasContrct extended forms model, and it is fully elaborated in [A.29].

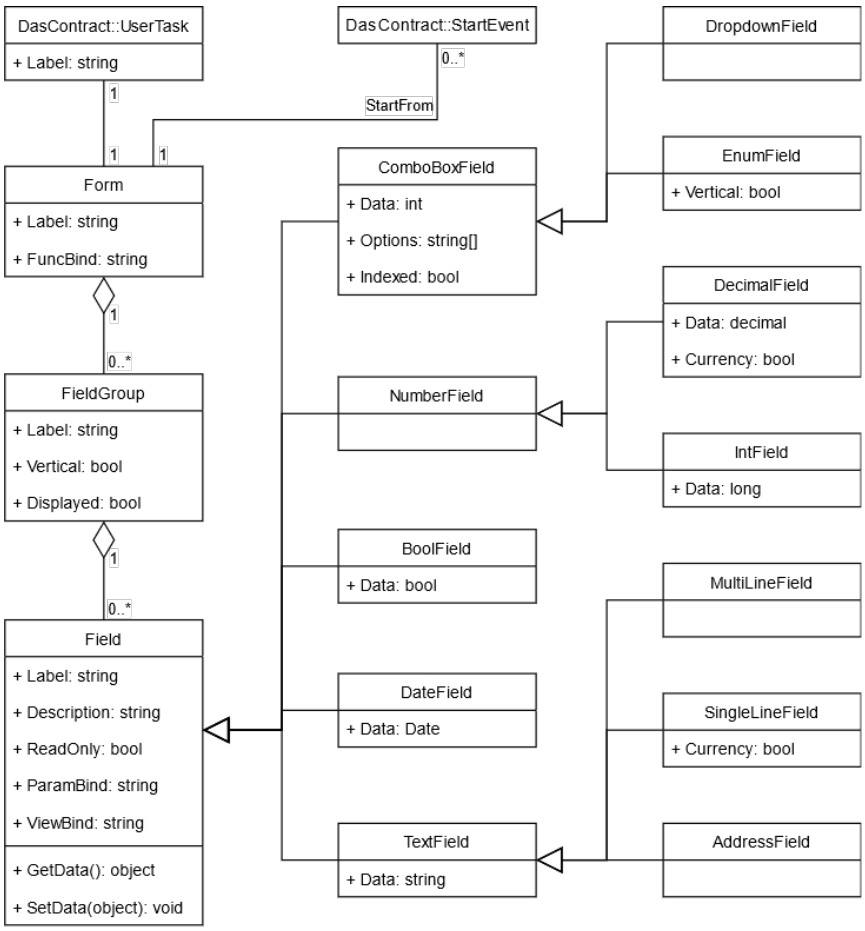


Figure 8.9: Extended forms metamodel [A.29]

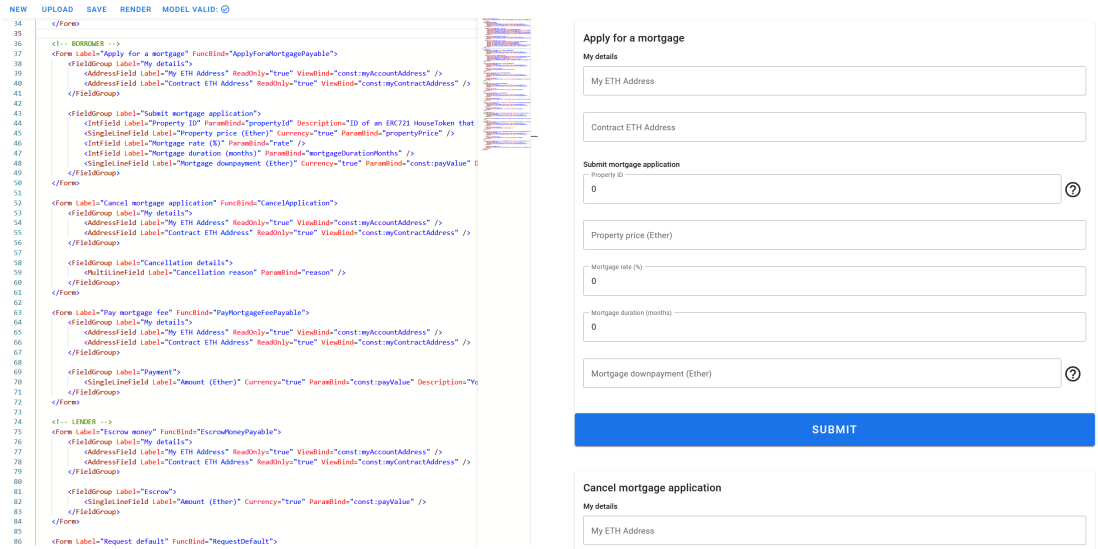


Figure 8.10: A screenshot of the extended forms editor [A.29]

## 8.5 Case Study: Mortgage

This section provides a case study to demonstrate the capabilities of the approach proposed in the previous section. A case of a decentralized mortgage that supports non-fungible tokens (ERC-721) to represent property ownership was created. This case study designs the process in a decentralized way, meaning that the property token is held in the smart contract as collateral. A scenario where the lenders can request a loan default is modeled. The complete DasContract model and the generated source code are published on Github under an MIT license [139].

### 8.5.1 Process Design

The following steps were taken while conducting this case study: 1) A DasContract model was created in a visual editor. 2) An Ethereum Solidity smart contract code was created by an algorithm described in Section 8.2. 3) A simulation of the generated smart contract was performed in the Remix [129] test environment.

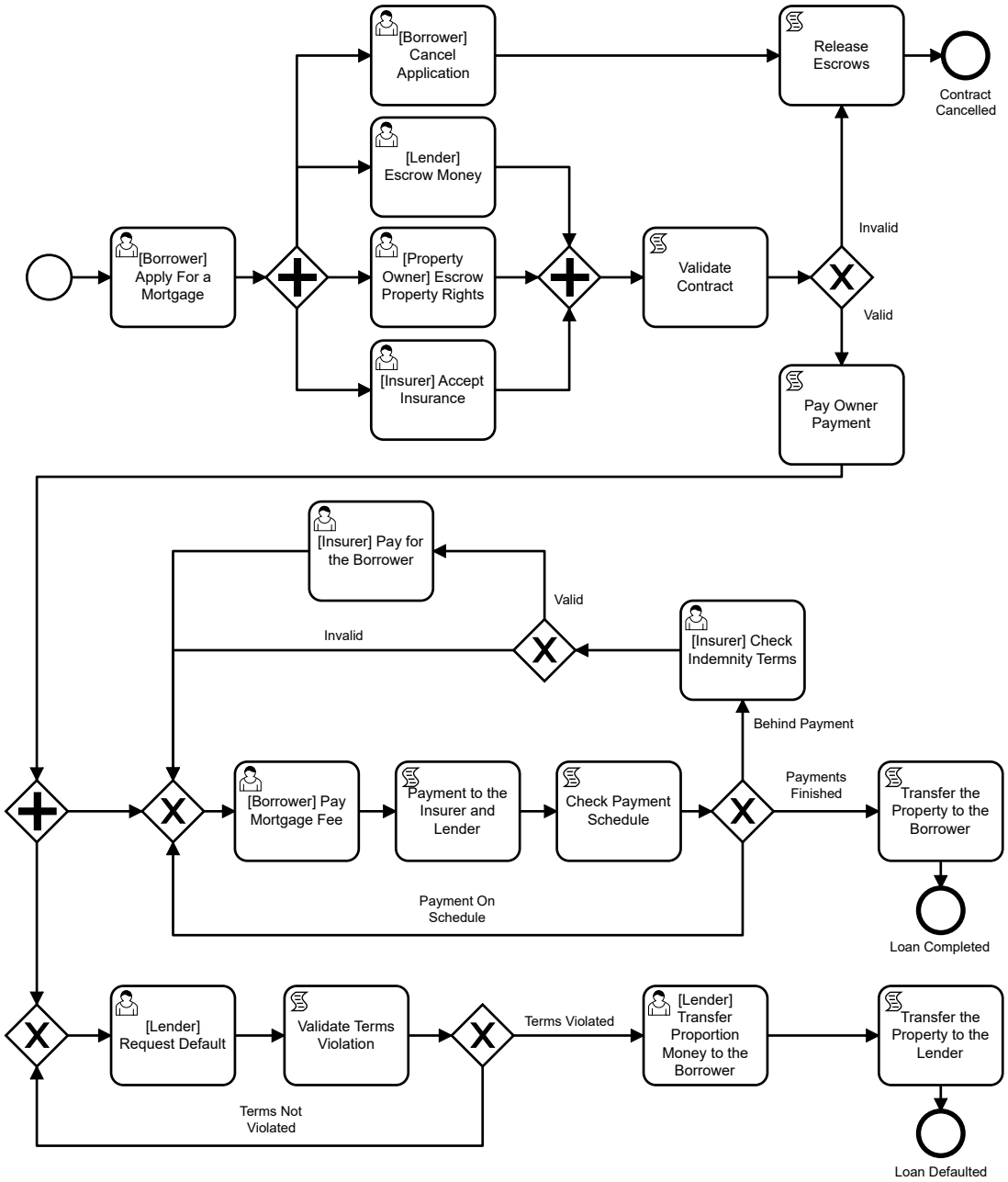


Figure 8.11: Mortgage process model

The process model of the proposed process can be seen in Figure 8.11. The process begins with a borrower applying for a mortgage. Three other parties – insurer, property owner, lender – must then confirm their involvement by carrying out their specific action. If any of them declines, or the borrower changes their mind, then the contract is deemed invalid, and any escrows that have been already transferred will be released back to their previous holders.

When all parties agree and the contract is successfully validated, the full payment to the owner is automatically released, ending their involvement in the contract. In the next phase of the contract, the borrower is tasked to pay the mortgage fees periodically. Those fees are then automatically distributed to the insurer and lender. The payment schedule is checked afterward, resulting in three possible scenarios. First, the payment is on schedule; the contracts state will return to "waiting" for another payment. Second, The payments are behind schedule; in this case, the insurer checks the indemnity terms, paying for the borrower if they are met. Third, the payments are finished, in which case the property is automatically transferred to the borrower, ending the contract.

Before the payments are fully finished, the lender is also at any time allowed to request for the borrower's default to check whether the borrower has not violated terms. If the terms have indeed been violated, then the lender will pay the proportion of money defined in the terms to the borrower. The property will then be transferred to the lender, ending the contract.

### 8.5.2 Execution

The described process was simulated using the Remix IDE. The test environment allows performing transactions from various addresses, enabling to simulate people interacting with the smart contract. In Figure 8.12 an image is shown from the model-driven off-chain environment based on the extended DasContract forms model described in Section 8.4. The forms model from which the form is rendered is shown in Figure 8.10.

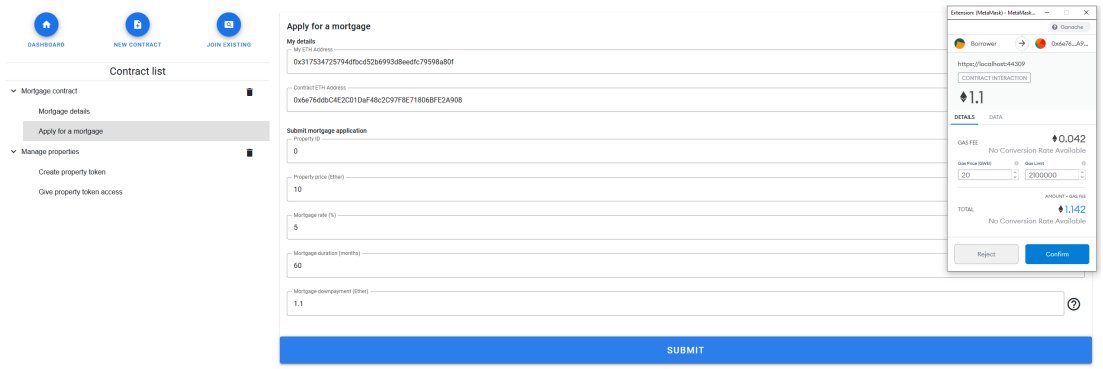


Figure 8.12: Off-chain interaction with the generated mortgage smart contract [A.29]

### 8.5.3 Summary

The limitations of the case study are the following. First, the mortgage is paid in a cryptocurrency, which is a very volatile asset. This can be addressed using stablecoins [103]. The second issue is that the case requires a cadastre of properties to be represented by a non-fungible token recognized in a country's legal context. This can be addressed by a legal binding of the token to property and mapping to an actual cadastre record using a blockchain oracle solution such as [42]. Third, the insurer is represented by a human oracle

and is not enforced by a code to pay. Therefore, the insurance terms would be needed to sign in a separate legal contract. This is already a part of the DasContract architecture described in Figure 8.3.

## 8.6 Case Study: EU Election

Since 1979, nine Elections of the European Parliament have occurred to date, once every five years according to the universal adult suffrage. From 2020, 704 Members of the European Parliament will be elected by more than 400 million eligible voters from 27 member states. For this reason, it is considered the second-largest democratic election in the world [70]. Each member state has its voting system, either by Preferential voting, Closed lists, or Single Transferable Vote. All can be combined with Multiple Constituents.

Further, each member state has its voting methods for citizens resident abroad and whether or not voting is compulsory. Each member state has different rules determining who can vote for and run as a Member of the European Parliament. Although this is not a very standardized process, the European Parliament is the only institution in the European Union, directly elected by the citizens. So, this must be a transparent, meddle-proof, usable, authenticated, accurate, and verifiable process [122].

The development and implementation of constitutional and legal provisions have been one of the engineering concerns. For instance, the Elections of the European Parliament have an estimated cost of 700 million euros. Blockchain has been the most promising technology when it comes to the electoral domain, seeing that per each vote, a new transaction, if valid, is added to the end of the Blockchain and remains there forever [25]. For this solution, no centralized authority is needed to approve the votes, and everyone agrees on the final tally as they can count the votes themselves, as anyone can verify that no votes were tampered with and no illegitimate votes were inserted.

This case study shows how to digitize the EU election process by following a method proposed in Section 8.1. A simplified version of the process was designed and tested on the Ethereum blockchain.

### 8.6.1 Process Design

The EU election process is very complex. The EU issues general guidelines on how country elections should look, and each country then implements its legislation to describe how the elections are done in a particular country. This means that each of the 27 EU countries does this process differently. To avoid this complexity, it was assumed that there is a unified voting process and each country votes according to one of the three voting systems – preferential voting, closed lists, and single transferable vote.

The modeled process is shown on Figure 8.13. It starts with an initiation of the elections where all the countries and their voting systems are initiated in the contract. After the initiation, political parties can register until a set deadline is expired. Later, candidates can register, and in the next step, the political parties approve the candidates. In

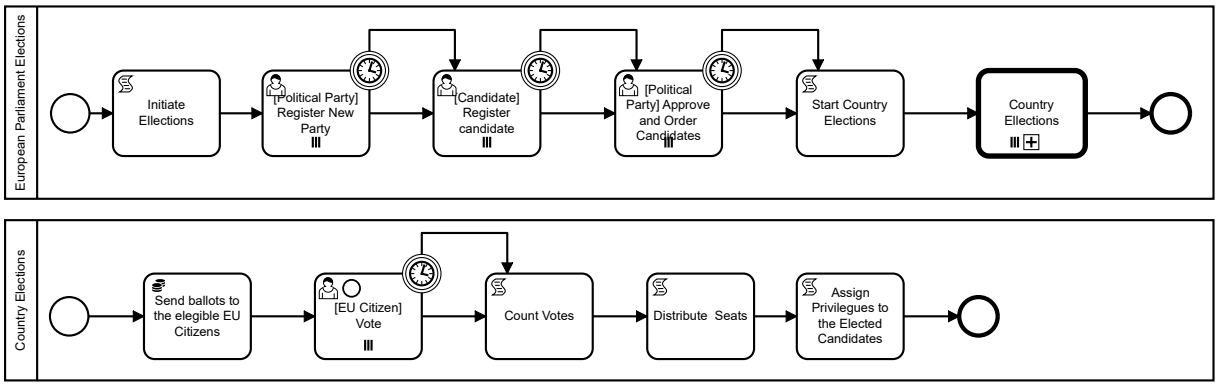


Figure 8.13: A DasContract process model of the EU elections

some countries, candidates without countries are allowed. After a deadline for approval is reached, the country elections for each initiated country are started in parallel as a subprocess.

The country elections subprocess starts by creating sending non-fungible tokens to eligible EU voters. The non-fungible tokens represent voting ballots to prevent a double vote. In the next step, the EU citizens can vote by sending their tokens to the election smart contract during the election days. After the voting is over, votes are counted, and seats are distributed according to the country's voting system. In the end, privileges are assigned to the selected candidates.

The process model also contains a validation and script task that is currently entered in the form of a solidity programming language. A domain-specific language is expected to be designed and used in the future.

The data model is shown on Figure 8.14. It only contains the essential information about the elections because the storage costs are high on public Blockchain because of a need for replication on all nodes and keeping all the history. A new concept added in this thesis is the possibility to specify tokens and enumerations in the data model. The voting token representing a ballot is specified as a non-fungible token, and to assure its uniqueness, it is associated with an individual voter identifier. The voter identifier represents a citizen's identity public key. However, the citizen's identification is outside of the scope of this thesis and is a subject of further research.

The forms model rendering for the process step *Vote* is shown on Figure 8.15. The DasContract currently generates only an on-chain validation code because the off-chain code is not supported on the Ethereum platform.

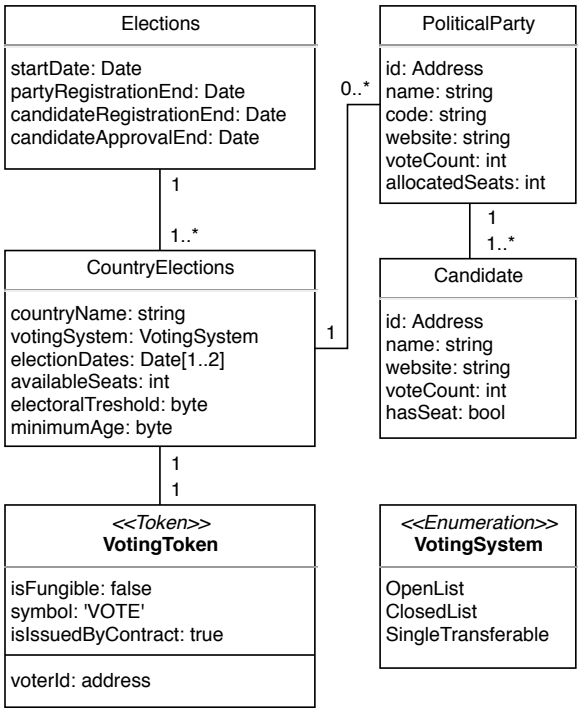


Figure 8.14: A DasContract data model of the EU elections


Figure 8.15: A user interface of the closed list vote in a blockchain wallet

### 8.6.2 Execution

The designed model was used during the execution step to generate a Solidity smart contract according to the MDE principles. The generation algorithm is available on GitHub under an open-source license [139].

An interesting example of the generated code shows an implementation of the vote function that accepts a non-fungible voting token and a voting choices cast by the citizen:

```
function vote(address[] memory votingChoices)
  public {
require(votingTokensContract
  .isEligibleToVote(msg.sender),
```

```
    "Voter not eligible to vote");
require(isVotingOpen(),
  "Voting is currently not allowed");
require(votingChoices.length > 0,
  "At least one cadidate must be chosen");

if(votingSystem == VotingSystem.ClosedList){
  require(
    politicalPartiesMap[votingChoices[0]]
      .exists,
    "Party address is invalid");
  politicalPartiesMap[votingChoices[0]]
    .voteCount++;
}
else if(votingSystem ==
  VotingSystem.OpenList){
    ...
}
else if(votingSystem ==
  VotingSystem.SingleTransferable){
    ...
}

votingTokensContract
  .transferVoteToken(msg.sender);
}
//The generated code was adjusted to improve
//it's readability.
```

The function's name is taken from the task name in a process model. The parameters of the vote functions are from the forms model associated with a user task. The address of a voting citizen and his voting ballot are not the parameters to prevent people from acting as someone else. The citizen's id is taken from the *msg* object that contains a verified public key by the original sender. Inside the function, the first step is to validate the person's eligibility to vote in the current election (E.g., he can be a minor and have no voting rights.). The second validation assures that the correct process step is selected. Finally, the votes are validated, so the citizen does not lose his ballot without voting for a valid candidate.

After the validations, a vote is counted according to the country voting system, and the citizen's vote is transferred back to the smart contract to prevent a double vote. The casting vote is counted but not forgotten due to blockchains' inherent history keeping. The Blockchain also guarantees that the function is processed one at a time, so the *voteCount++* is safe to be used even when it is called from multiple sources in parallel.

In the end, the generated source code was simulated in the Ethereum Remix simulation environment. A screenshot from the simulation showing the vote function is shown in

Figure 8.16.

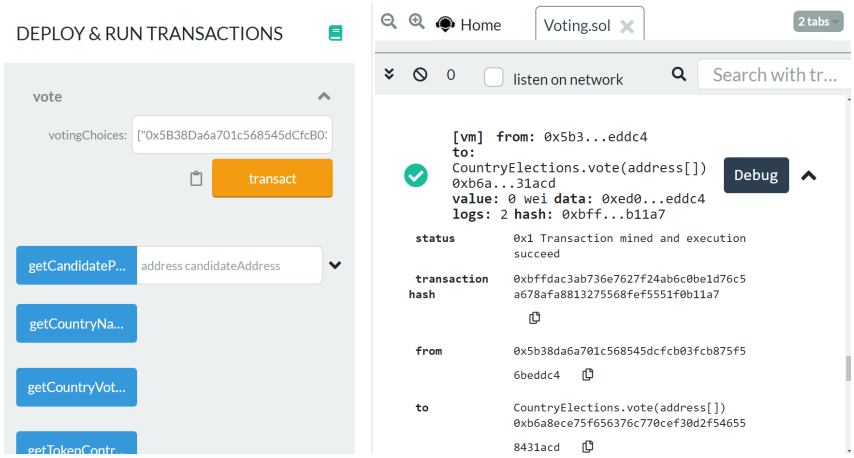


Figure 8.16: A screenshot from Remix simulation

### 8.6.3 Summary

Currently, the election process can be designed, simulated, and executed in blockchain technology. However, the public blockchain implementation Ethereum capacity only allows processing up to fifteen transactions per seconds [9] for all of its users, and an average transaction price as of September 16th, 2020 is 4.301 USD [170]. The two reasons alone would make it impossible to run the real EU parliament elections on the Ethereum blockchain. It is expected that the transaction prices will go lower in the future, and the transactions per second will increase due to the implementation of the proof of stake consensus algorithm and second-layer scaling solutions. These scaling limitations do not apply to private blockchain solutions such as Hyperledger Fabric [153], so it would be possible to execute the EU election process there. However, the private blockchains may not guarantee the same security properties as the public blockchains because a handful of selected operators runs the network.

Following the proposed method allowed us to model the EU elections case in a visual editor using the DasContract language. The DasContract only allowed for concepts that are valid in Blockchain. Furthermore, the data model was expressed, and a custom non-fungible token was designed. It was possible to test the process model's behavior during the design using standardized BPMN simulation tools.

A Solidity smart contract was generated using an open-source algorithm during the execution phase. The generated code contains 365 lines, and it was executed in the Remix environment. Due to Ethereum transaction fees, it would not make economic sense to run it on millions of users.

Overall, the proposed approach makes the digitalization of processes in the blockchain environment easier as it is based on a standardized process modeling notation. The generation of smart contract code reduces programming errors. However, only Ethereum smart

contracts can be generated, but the DasContract model seems to be transferable to other blockchain implementations.

## 8.7 Limitations

There are the following limitations to this approach, and they should be addressed in future research. First, the script tasks and a validation logic of user tasks still need to be expressed in Solidity language. An implementation-independent DSL for such expressions should be added to the design of the language. Second, the language only contains support for receiving tokens. Support to issue both fungible and non-fungible tokens would be required. Third, it is not clear how would people authenticate themselves and prove their roles outside of the standard smart contract wallet. A way to support decentralized identity standards such as W3C DID should be explored. Finally, according to Gartner [54], the public blockchains are still immature for mass adoption mostly because of low transaction processing capability and high cost and, therefore, only minimal applications such as initial coin offerings, escrows, and decentralized finance are currently possible. The major limitation of this approach remains the immaturity of available blockchain implementations. According to the Gartner [54], the blockchain technology is very immature to support most of the potential use cases, and there is still a tremendous amount of research, implementation, and adoption to be done.

## 8.8 Related Research

As for the modelling law for the purposes of executing it in a blockchain smart contract, we are not aware of any visual language to describe all aspects of the smart contracts - process, data structures, and actions. However, there are already existing approaches that use BPMN [114] or Blockly [57]. An interesting project is Marlowe [93] which builds a domain specific language for the financial domain.

A very similar research is done by the Caterpillar project [125, 97] that is creating a BPM engine in the Solidity language. The main difference between the approaches is that the DasContract generates the logic of the model into the smart contract where the Caterpillar interprets it. The goal of the DasContract is to provide a decentralized way to conduct contracts between people, companies, and governments in a blockchain implementation independent way. The Caterpillar is currently bound to the Solidity programming language.

Other approaches provide a graphic environment to visually compose smart contract code based on Blockly [57] such as [56]. There is also another approach that proposes a domain specific language for definition of financial smart contracts called Marlowe [93]. The approach presented in this paper does not represent the script part in the visual format; it instead focuses on modeling data, processes, and forms that is not supported by Blockly-based approaches.

In [168] was proposed a tool to support inter-organizational processes through Blockchain technology. To ensure that the joint process is correctly executed, the control flow and business logic of inter-organizational business processes are compiled from the processes models into Smart Contracts. Weber et al. developed a technique to integrate Blockchain into the choreography of processes to maintain trust, employing triggers and web services. By storing the status of process execution across all involved participants, as well as to coordinate the collaborative business process execution in the Blockchain. The validation was made against the ability to distinguish between conforming and non-conforming traces. [52] presented an optimization in regards to the already presented paper [168]. To compile BPMN models into a Smart Contract in Solidity Language, the BPMN model is first translated into a reduced Petri Net. Only after this first step, the reduced Petri is compiled into a Solidity Smart Contract. Compared to [168], [52] managed to decrease the amount of paid resources and achieve higher throughput.

## 8.9 Chapter Summary

In this chapter, we introduced a vision of contracts between people, companies, and legal authorities that can be partially automated and executed in blockchain smart contracts. It was argued that the proposed concept could significantly impact how contracts are conducted. To pursue the goal, a visual domain-specific language for modeling blockchain smart contracts was introduced and demonstrated in two case studies.

# Part VI

# Evaluation and Conclusion

Chapter 9

# Evaluation and Contribution

In this chapter, the contributions of this thesis in the form of the DSR artifacts, publications, and related supervised theses are presented in Section 9.1. The research objective and the research questions are evaluated and answered in Section 9.2. Finally, the chapter is summarized in Section 9.4.

## 9.1 Contributions of the Dissertation Thesis

This section presents all relevant contributions of this dissertation thesis. First, the research artifacts are presented. Second, an overview of publications is provided. Finally, the contributing supervised theses are outlined.

### 9.1.1 Research Artifacts

This section provides an overview of all research artifacts that were output from the DSR research cycles. The artifacts are divided into four types: formalization (F), method (M), experiment (E), and case study (C). An overview of all research artifacts is shown in Figure 9.1. The overview shows the contributions in the context of the compliance management domain. In light blue, there are contributions to centralized compliance management. Purple shows contributions to decentralized compliance management.

The following list presents a summary of the artifacts together with the DSR cycle in which they were created. The main author's contribution is also discussed in cases where multiple parties collaborated.

- **F1 – FAR Ontology (Cycle 1.1)** – The FAR ontology (Section 4.1) introduces definitions of facts, acts, and rules suitable for software system execution. It is the foundation upon which the DEMO machine is built.
- **F2 – DEMO Machine (Cycle 1.1)** – Is an execution language to simulate the DEMO models and enables the creation of a BPMS based on DEMO methodology. The DEMO machine is presented in Section 4.2.

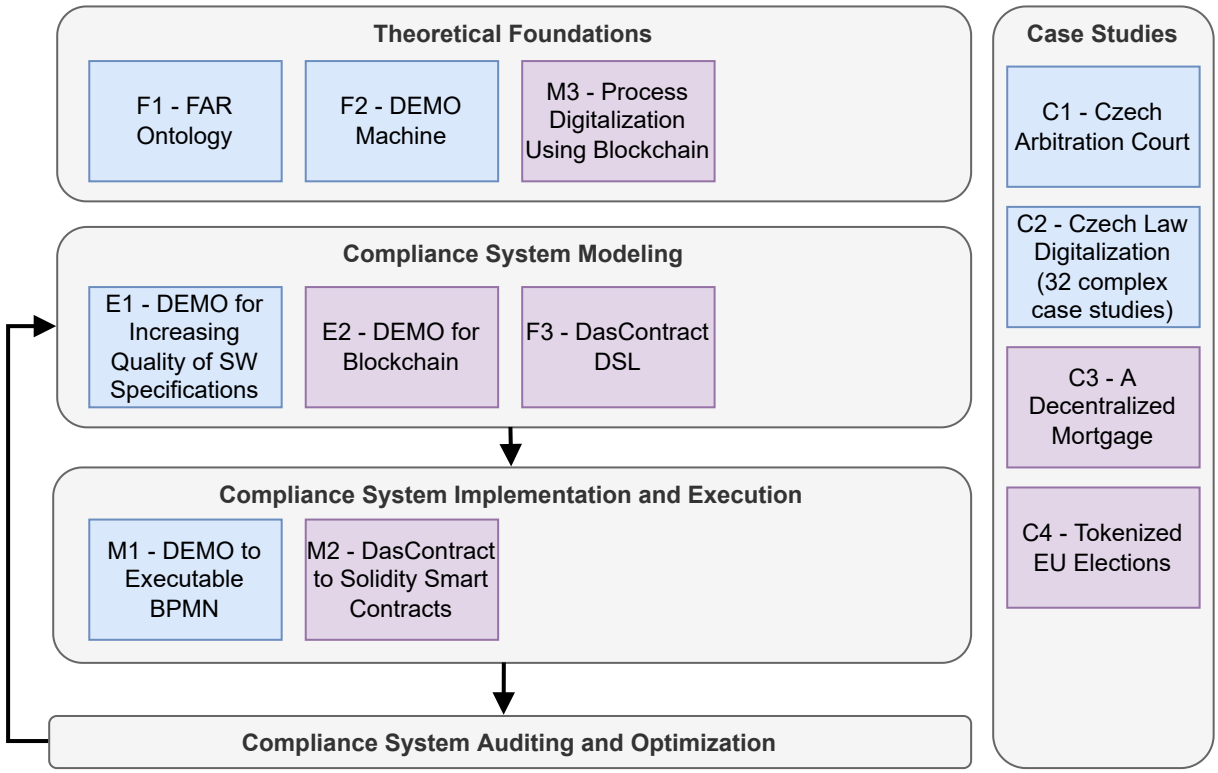


Figure 9.1: Research artifacts

- **F3 – DasContract DSL (Cycle 2.2)** – Is a domain-specific language that allows modeling of the blockchain smart contracts in an implementation-independent way. The language is presented in section 8.2. The artifact M2 shows a way to generate the Ethereum Solidity smart contracts from the DasContract models. A proof of concept implementation of the DasContract editor and an environment to generate an executable blockchain smart contract was developed and published on GitHub [139]. The proof of concept implementation was done to validate the DasContract DSL and M2, and many supervised theses contributed to the implementation. The main author supervised the development of the proof of concept.
- **M1 – DEMO to Executable BPMN (Cycle 1.2)** – This method provides a way to generate executable BPMN models from the DEMO models (Chapter 5). The execution semantics are based on the DEMO machine. O. Mráz described the method and the main author of this thesis contributed a detailed explanation of the execution of the DEMO models according to the DEMO machine.
- **M2 – DasContract to Solidity Smart Contracts (Cycle 2.2)** – Describes a way to generate Ethereum Solidity smart contracts from DasContract models (Section 8.3). The main author contributed supervision and guidance towards execution of the DasContract models in a blockchain-independent way, and the co-author [A.32] contributed with a specific implementation in the Ethereum Solidity.

- **M3 – Process Digitalization Using Blockchain (Cycle 2.2)** – is a part of our proposed approach to decentralized compliance presented in Section 8.1. This approach and the method were a contribution of the main author. The co-authors contributed with specific domain knowledge required for the case studies and took part in the modeling and implementation of the case studies.

- **E1 – DEMO for Increasing Quality of SW Specifications (Main Cycle 1)** – this experiment was designed to empirically explore capturing complex process requirements to digitize them using a software system. In the literature, we found only elementary examples. However, large and complex process descriptions are used in compliance management practice. The domain of procedural law was selected because the process descriptions are publicly available and very complex. The results (Section 6.4) showed that applying DEMO to modeling process-based software requirements helps to identify missing process acts and actors. All case studies were published on GitHub [96, 144, 143]. The main author conducted the experiment, provided extensive tutoring to the participants, and created a coursebook focused on applying DEMO [A.12]. The participants of the experiment were students at the Czech Technical University. The experiment consisted of 32 case studies where 115 276 words of the legal text were analyzed in approximately 2 440 hours. The outputs of this experiment are artifacts C1 and C2.

- **E2 – DEMO for Blockchain (Cycle 2.1)** – this experiment was conducted to explore the role of blockchain technology in EE theories (Section 7.1). The investigation started as a supervised thesis [A.21] and resulted in a publication [A.4]. Because the publication was well received, the outcomes of the publications served as business requirements for the DSR cycle 2.2, where the F3, M2, and M3 were created.

- **C1 – Czech Arbitration Court (Main Cycle 1)** – this was the first case study conducted as part of E1 (Section 6.2). The first iteration of the case study was done in a supervised thesis ([A.24]) and later improved together with the main author for publication [96]. The study was used as a reference example for C2.

- **C2 – Czech Law Digitalization (31 Case Studies) (Main Cycle 1)** – were the case studies conducted by students under the supervision and lecturing of the main author as part of the Enterprise modeling class at the Czech Technical University (Section 6.3). To achieve a great quality of the studies, only 22 of the 31 case studies were manually selected for the final dataset published on GitHub [144, 143].

- **C3 – A Decentralized Mortgage (Main Cycle 2)** – is a first case study to demonstrate the DasContract DSL(Section 8.5). The complete DasContract model and the generated source code are published on Github [139]. The main author created the decentralized mortgage case in DasContract DSL. The collaborators contributed to implementing and testing the generated contract and analyzing the mortgage domain in the Czech Republic.

- **C4 – Tokenized EU Elections (Main Cycle 2)** – this case study (Section 8.6) was made as a collaboration with a Portuguese master's degree student who was interested in the DasContract research. The parties collaborated on creating this case study and published the results in [A.7].

### 9.1.2 Publications

This section presents all peer-reviewed publications related to the research in Table 9.1. The related DSR artifacts and DSR cycles are included as well. A complete list of citations is provided in Part VII. The individual contributions of the main author were discussed in Section 9.1.1.

Table 9.1: Overview of our publications

| **Publication** | **Artefacts** | **DSR Cycles** |
|---|---|---|
| Skotnica M.; van Kervel S.J.H.; Pergl R. Towards the Ontological Foundations for the Software Executable DEMO Action and Fact Models [A.1] | F1 – FAR Ontology | Cycle 1.1 |
| Mráz O.; Náplava P.; Pergl R.; Skotnica M. Converting DEMO PSI Transaction Pattern into BPMN: A Complete Method [A.2] | M1 – DEMO to BPMN | Cycle 1.2 |
| Skotnica M.; van Kervel S.J.H.; Pergl R. A DEMO Machine - A Formal Foundation for Execution of DEMO Models [A.3] | F2 – DEMO Machine | Cycle 1.1 |
| Hornáčková B.; Skotnica M.; Pergl R. Exploring a Role of Blockchain Smart Contracts in Enterprise Engineering [A.4] | E2 – DEMO for Blockchain | Cycle 2.1 |
| Skotnica M.; Pergl R. Das Contract - A Visual Domain Specific Language for Modeling Blockchain Smart Contracts [A.5] | F3 – DasContract DSL | Cycle 2.2 |
| Skotnica M.; Klicpera J.; Pergl R. Towards model-driven smart contract systems - code generation and improving expressivity of smart contract modeling [A.6] | M2 – DasContract to Solidity<br>C3 – A Decentralized Mortgage | Cycle 2<br>Cycle 2.2 |
| Skotnica M.; Aparício M.; Pergl R.; Guerreiro S. Process digitalization using blockchain: EU parliament elections case study [A.7] | M3 – Blockchain Process Digitalization<br>C4 – EU Elections | Cycle 2 |

### 9.1.3 Supervised Theses

This section presents an overview of the supervised student theses related to the research presented in this thesis. The main author supervised 22 student theses relevant to this

thesis, and 3 of them received the dean's award. Table Table 9.2 presents all theses and their relation to the DSR research cycles and research artifacts. In cases where the theses were related to the research but had results not presented in this thesis, we marked the related artifact as N/A. Some of the student theses followed with peer-reviewed publications presented in the previous section. The individual contributions of the main author were discussed in Section 9.1.1. The student works usually explored new and interesting possibilities in the research domain or helped with the implementation and tooling for the concepts proposed in this thesis.

Table 9.2: Overview of supervised publications related to the research

| **Supervised Thesis** | **Related DSR Cycle** | **Related Artefact** |
|---|---|---|
| Buša R. Designing WYSIWYG Web Forms [A.13] | Cycle 2 | M2 – DasContract |
| Nymsa P. Mobile Enterprise Architecture Process Analytic Tool Based on the DEMO Methodology [A.14] | Cycle 1 | N/A |
| Ančinec P. Open-source DEMO Construction and Process Model Designer [A.15] | Cycle 1 | N/A |
| Bydžovský T. A State Management in Multi-client Single Page Web Applications [A.16] | Cycle 1 | N/A |
| Drozdík M. Open-Source Legal Process Designer in .NET Blazor [A.17] | Cycle 2 | F3 – DasContract DSL |
| Krbilová K. Process Mining in Finance Domain [A.18] | Cycle 1 | N/A |
| Šelder O. Generating Plutus Smart Contracts from DEMO Process Models [A.19] | Cycle 2<br>Cycle 2.2 | M2 - DasContract to Solidity |
| Lassaková M. Law Modelling Using BPMN and DEMO [A.24] | Cycle 1 | C1 - Czech Arbitration Court |
| Hornáčková B. Using Blockchain Smart Contracts in the DEMO Methodology [A.21] | Cycle 2.1 | E2 - DEMO for Blockchain |
| Jančovičová B. Next Generation Methods for Development of Enterprise Information Systems [A.22] | Cycle 1 | N/A |
| Lang M. WebAssembly Approach to Client-side Web Development using Blazor Framework [94] | Cycle 1 | N/A |
| Mikeš S. Evolvability of Business Process Models [A.25] | Cycle 1 | N/A |
| Mužák M. Model-Driven Approach to Governance, Risk, and Compliance Systems Development [A.26] | Cycle 1 | N/A |

Table 9.2: Overview of supervised publications related to the research

| **Supervised Thesis** | **Related DSR Cycle** | **Related Artefact** |
|---|---|---|
| Frait J. Generating Ethereum Smart Contracts from DasContract Language [A.27] | Cycle 2.2 | M2 - DasContract to Solidity |
| Bydžovský T. Decentralized Identity in DasContract Decentralized Applications [A.28] | Cycle 2 | F3 - DasContract DSL |
| Ančinec P. Domain-Specific Languages for Offchain UI in Decentralized Applications [A.29] | Cycle 2.2 | F3 - DasContract DSL |
| Urbánek Š. Exploring the use of Blockchain Smart Contract in the E-Commerce [A.30] | Cycle 2.2 | F3 - DasContract DSL |
| Škrabal M. Use Cases for Decentralized Identity [A.20] | Cycle 2 | N/A |
| Drozdík M. Generation of Plutus Smart Contracts from DasContract models [A.31] | Cycle 2.2 | M2 - DasContract to Solidity |
| Klicpera J. Client-Side Application Development Using Blazor Framework - a Blockchain Smart Contract Designer Case Study [A.32] | Cycle 2.2 | F3 - DasContract DSL |
| Krbilová K. Blockchain Smart Contracts in Public Sector [A.33] | Cycle 2 | F3 - DasContract DSL |
| Šelder O. Business Rules in Blockchain Smart Contracts [A.34] | Cycle 2.2 | F3 - DasContract DSL |

## 9.2 Evaluation of the Research Objective and Research Questions

This section evaluates the research objective that was set to: "Investigate the gaps between the DEMO methodology and its application in the design of centralized and decentralized compliance management systems. For some of the identified gaps, propose artefacts that could contribute to filling them. Finally, provide case studies of the proposed approaches that resemble real-world use cases."

To answer this research objective, two main research questions and four sub-research questions were proposed. To answer the research questions, the research design (Figure 9.2) and research strategy were presented in Chapter 3 according to the design science research methodology. The resulting research artifacts were presented in Section 9.1.1 and are mapped to the DSR cycles and research questions in Table 9.3. An extensive literature review of the state-of-the-art was made in Part II. The centralized DSR cycle results were presented in Part IV. The decentralized DSR cycle results were presented in Part V. Finally, the research questions are answered in this section.

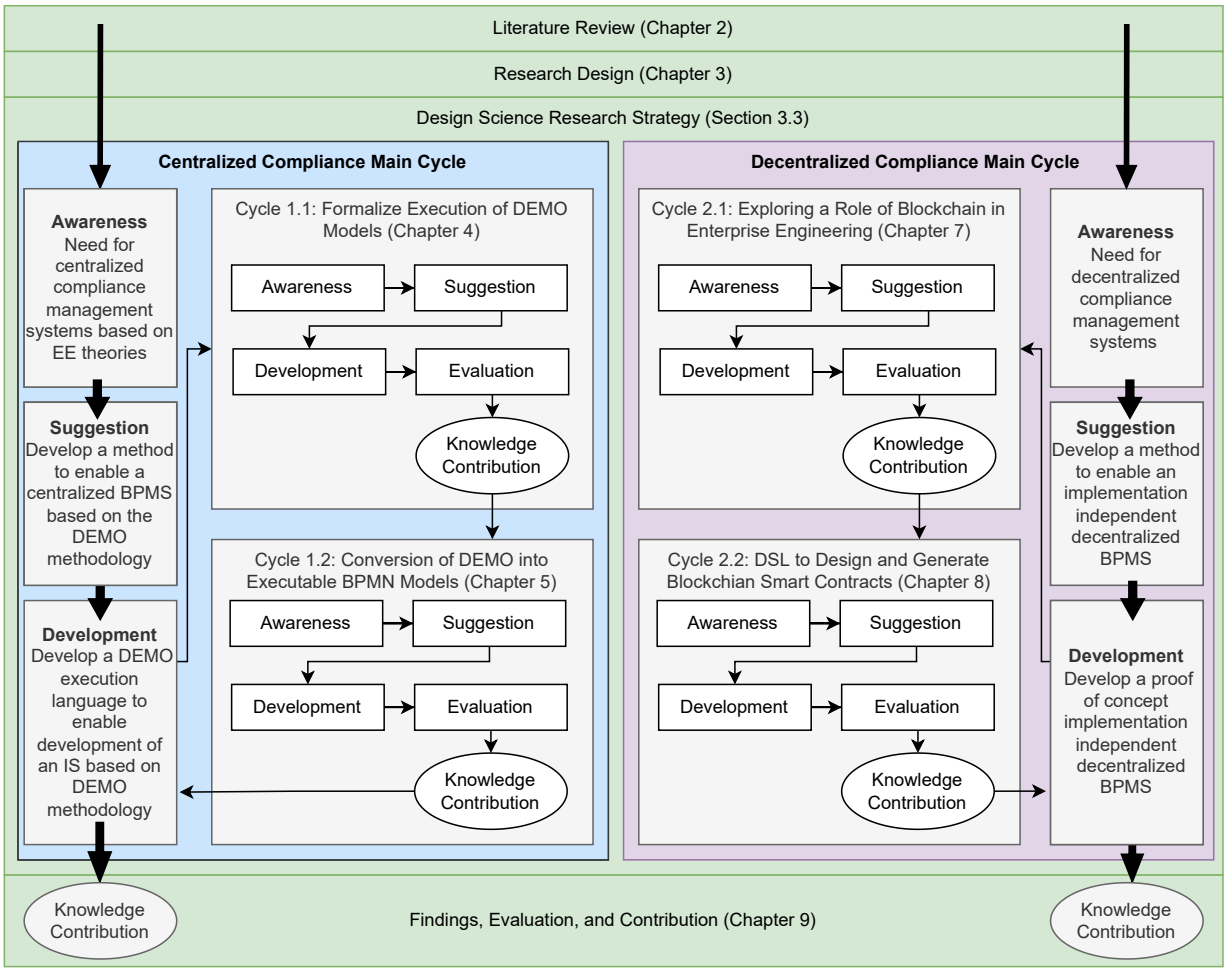


Figure 9.2: Our research strategy

### 9.2.1 Research Question 1

Research question 1 was formulated as: "How to design software systems to support business process management requirements based on EE theories and DEMO methodology?"

This research question was answered in the main DSR cycle 1 for centralized compliance management and presented in Part IV. The research question contains two sub-research questions that are answered in Section 9.2.1.1 and Section 9.2.1.2. As described in Section 3.2.1, the business requirements for this DSR cycle came from cooperation with a professional company. The resulting artifacts are summarized in Section 9.1.1 and mapped to the research questions in Table 9.3.

To summarize the answer to the research question: The software systems to support BPM requirements based on EE theories and the DEMO methodology can be designed by using the DEMO machine execution language. To demonstrate the feasibility of this approach, a practical implementation was created in cooperation with a professional company and went to professional production (Section 9.3). Furthermore, experiment E1 was

Table 9.3: Mapping of our research questions, DSR cycles, and artefacts

| Research Question | Centralized Cycle | Cycle 1.1 Chapter 4 | Cycle 1.2 Chapter 5 | Decentralized Cycle | Cycle 2.1 Chapter 7 | Cycle 2.2 Chapter 8 |
|---|---|---|---|---|---|---|
| RQ 1 | C1,C2,E1 | | | | | |
| SRQ 1 | F1,F2 | F1,F2 | | | | |
| SRQ 2 | M1 | | M1 | | | |
| RQ 2 | | | | C3,C4 | | |
| SRQ 3 | | | | | E2 | |
| SRQ 4 | | | | M2,M3,F3 | | M2,M3,F3 |

conducted to show that the EE theories and DEMO methodology can increase the quality of process-based software requirements in the domain of procedural law (Section 6.1).

The main limitation of using the EE theories and DEMO methodology to design software systems to support BPM requirements is that the DEMO methodology is inherently limited in its conceptualization possibilities. It is only applicable for ontological clarification, assigning responsibilities, and modeling objects with their properties (the FM). It does not help with any implementation-specific or technology-specific concerns, as they are beyond its scope. However, complementing it with other methodologies and notations is expected to be possible. Next, applying the DEMO methodology requires nontrivial knowledge and practice, demonstrated in our experiment E1 (Section 6.1). Performing this manually requires considerable elaboration.

#### 9.2.1.1 Sub-Research Question 1

The sub-research question 1 was formulated as: "How to design software systems to support business process management requirements based on EE theories and DEMO methodology?"

This research question was answered in DSR cycle 1.1 by proposing FAR Ontology (F1) and DEMO machine (F2) in Chapter 4. The formalizations were published in peer-reviewed publications [A.1, A.3]. The practical implementation of the formalizations was done in a professional software solution and went to professional production as described in Section 9.3.

The main limitation of the execution language is that it is designed to support only the full transaction axiom according to the DEMO theory. In software systems, the full transaction axiom is rarely needed, and a subset of the transaction axiom would be enough. This would be solved by adding more flexibility in the transaction axiom state machine (Figure 4.1). However, the issue can be resolved in the presented formalization by setting action rules for the unused transaction acts.

#### 9.2.1.2 Sub-Research Question 2

The sub-research question 2 was formulated as: "How should DEMO models be transformed into BPMN models?" The research question was answered in DSR cycle 1.2 by proposing a method to convert DEMO models into executable BPMN models (M1) in Chapter 5. The method was published in a peer-review publication [A.2].

The main limitation of this approach is that the resulting BPMN diagrams are too large. This is because the full DEMO transaction axiom is generated for each transaction. Therefore, this approach should only serve as a blueprint for the transformation between DEMO and BPMN, and the resulting BPMN should be manually adjusted according to the best practices such as [135, 132]. Follow-up work was done in [155]; however, the main limitation still remains.

### 9.2.2 Research Question 2

The research question 2 was formulated as: "How to digitize business processes using blockchain smart contracts in a methodical way and eliminate programming errors while avoiding a dependency on a particular blockchain implementation?"

This research question was answered in the main DSR cycle 2 for decentralized compliance management and presented in Part V. The research question contains two sub-research questions that are answered in Section 9.2.2.1 and Section 9.2.2.2. The resulting artefacts are summarized in Section 9.1.1 and mapped to the research questions in Table 9.3.

To summarize the answer to the research question: The business processes can be digitized using blockchain smart contracts in a methodical way by modeling the smart contract in the proposed DasContract DSL F3 (Section 8.2) following the proposed approach and method M3 (Section 8.1). A proof of concept implementation was created based on the proposed formalism F3 and method M3 as part of supervised student theses (Section 9.1.3). Further validation of this approach was done on two case-studies – a decentralized mortgage C3 (Section 8.5) and tokenized EU elections C4 (Section 8.6). Further case studies were created as part of supervised student theses in [A.30, A.34, A.33].

The main limitation of our approach is that we could not empirically verify the elimination of the programming errors. Moreover, avoiding a dependency on a particular blockchain implementation is only partial, as mentioned in Section 9.2.2.2.

#### 9.2.2.1 Sub-Research Question 3

The sub-research question 3 was formulated as: "How can blockchain smart contracts be used in the implementation of a software system based on DEMO methodology?" The research question was answered in DSR cycle 2.1 and presented in Chapter 7. The results were also published in a peer-review publication [A.4].

We have answered sub-research question 3 by proposing two ways of using smart contracts in the DEMO methodology in the context of EIS. Such integration would bring blockchain benefits to process execution, namely secured and trustless storage of data and

immutable transaction execution. This might bring a new way of looking at transactions with external actors, where the blockchain can serve as trustless coordination of the operation and a notarized data source. The application of the DEMO methodology to the business processes behind smart contracts may bring insight and overview to the whole operation. This may help to reduce unwanted states, prevent errors and improve security which is crucial for smart contracts because they represent valuable assets. Further, it may serve as a basis for creating well-designed Dapps and DAOs.

The major limitation of this approach was that the DEMO methodology applies only to business processes involving human cooperation and co-production because it is based on Habermas's sociological theory of communicative acts [63]. If the target information system is not expected to be controlled by human actors, using modeling languages described in Section 2.2.1 and Section 2.2.2 is preferable. However, the existing formal languages do not contain the ontological capability to express blockchain-specific constructs. Therefore, this major limitation led to the formulation of the sub-research question 4.

#### 9.2.2.2 Sub-Research Question 4

The sub-research question 4 was formulated as: "Is it feasible to generate blockchain smart contracts from a high-level modelling language in a automated methodological way?" The research question was answered in DSR cycle 2.2 and presented in Chapter 8. The results were also published in peer-review publications [A.5, A.6].

Yes, it is feasible to generate blockchain by modeling the smart contract in the proposed DasContract DSL (Section 8.2) following the proposed approach and method M3 (Section 8.1). And then use the proposed method M2 to convert the DasContract model into the Ethereum Solidity smart contract (Section 8.3).

The major limitation of this approach remains the immaturity of available blockchain implementations. According to Gartner [54], blockchain technology is very immature to support most potential use cases, and there is still a tremendous amount of research, implementation, and adoption was done. The second limitation of our approach is that the DasContract model only allows us to express the procedural part of the contract and the user input. The generated needs to be extended with a blockchain-specific source code to work on a particular blockchain implementation.

## 9.3 Application of the Contributions

This section provides an overview of the theoretical and practical application of our contributions presented in Section 9.1.

The most important application of our research came from our cooperation with a professional company ForMetis Consultants BV [50], during the main centralized DSR cycle. The design and development of the FAR ontology and DEMO machine were consulted with the company and contributed to the creation of a proprietary ForMetis DEMO engine. According to van Kervel's (director of ForMetis) letter of support that is included in our

PhD thesis submission: "This software engine has been put in a professional production at several companies where the unique precise implementation of complex business processes has been proven." Due to the proprietary nature of the ForMetis DEMO engine and the privacy of the companies where it was implemented, we are not allowed to publish more information about the projects.

The FAR ontology was used to allow modeling of REA compliant accounting systems in [78]: "The FAR ontology enables the CC-CP model to utilize all communication facts and any logic aggregated facts. The DEMO CC-CP model captures all the facts relevant to the REA exchange process, and even other facts that are produced by REA information and business events." The paper then concludes with: "All relevant real-world phenomena must be well captured by the DEMO CC-CP model; otherwise it is impossible to devise a working REA compliant accounting system." [78].

Our experiment in the domain of procedural law (Section 6.1) showed that applying the EE theories and DEMO methodology could lead to an increased quality of process-based software requirements. Compared with similar approaches (Section 6.4.5), our experiment provides a large dataset published on this topic with 115 276 words. Other known methods showed only smaller examples with up to 300 words. The proposed method provides a clear method for determining missing process steps and actor roles while offering strong guidelines for complexity management. This was not observed in similar approaches. All 23 case studies were published on GitHub [96, 144, 143].

A proof of concept implementation of the DasContract editor and an environment to generate an executable blockchain smart contract was developed and published on GitHub [139]. Two complex case studies were created – a decentralized mortgage (Section 8.5) and tokenized EU elections (Section 8.6) as a validation of the proposed concept. Further case studies were created as part of supervised student theses in [A.30, A.34, A.33]. The published results should serve as guidance for further research in the decentralized compliance management systems domain. The interest in research in this area can be observed in a number of citations of our peer-reviewed papers [A.4, A.5, A.6, A.7] presented in Part VII.

## 9.4 Chapter Summary

In this chapter, all contributions to our research were presented as research artifacts, peer-reviewed publications, and supervised theses. The original contribution of the main author was clarified. The evaluation of the research objective and research questions was presented with a summary of the main limitations. Finally, an application of the contributions was mentioned.

Chapter 10

# Conclusions

In this chapter, the whole thesis is summarized and the future work proposed.

## 10.1 Summary

In the introduction (Chapter 1) the domain of compliance management was introduced, and the main challenges were pointed out. Digitalization of compliance management efforts is desirable, and the discipline of EE can improve state of the art. Furthermore, a novel field of decentralized compliance management based on blockchain technology is emerging and needs deeper exploration. We narrowed our scope to the business process aspect of the compliance management domain in formulating a research problem. Then proceeded to formulate the research objective and research questions. An overview of the research methodology and research strategy was presented.

Chapter 2 provided an overview of the most important theories and methods relevant to the research. An overview of the compliance management approaches was presented together with an overview how to model the domain and support with software. An introduction to the most relevant EE theories, DEMO methodology, and the OER method was made with a discussion about the ontological qualities of DEMO models. Comprehensive guidance on how to read the DEMO models was made for readers unfamiliar with the methodology.

In Chapter 3, the research strategy and research design to answer the research objective and research questions were presented. Our research uses the DSR methodology adopted for IS research according to best practices for PhD theses published by other researchers. According to the DSR methodology, a rearch strategy for our research is presented. The research strategy consists of two main DSR cycles, one for the centralized compliance systems and the second for the decentralized compliance systems. The assumptions, scope, and limitations of our research were also presented.

Part IV presents the most important results from our centralized compliance DSR research cycle. The main focus was finding and bridging the gaps between the EE theories and their software implementation. A professional company provided the business require-

ment for this DSR cycle and identified the largest gap as a need for an execution language for DEMO models. To bridge this gap, the FAR ontology and DEMO machine formalizations were developed and presented in Chapter 4. The professional company further used the proposed formalizations to develop a software system successfully deployed to its customers.

An alternative approach to bridge the gap between the EE theories and their software implementation was proposing a method to convert DEMO models to BPMN models. This method was presented in Chapter 5. The method serves as a blueprint for how to proceed with the execution of DEMO models in state-of-the-art software systems.

In Chapter 6, an experiment to apply the EE theories to improve the quality of process-based software requirements. The experiment consisted of 32 case studies where 115 276 words of the legal text were analyzed in approximately 2 440 hours. The evaluation of the case studies revealed missing process steps ($\overline{80.10}\%$) and actor roles. Only 23 of the 32 case studies were manually selected, published on GitHub [96, 144, 143], and used in the evaluation.

Part V presents the most important results from our decentralized compliance DSR cycle. The main focus was on formalizing a language DasContract to model decentralized compliance processes and then provide support to generate blockchain smart contracts from the developed formalization. A proof of concept implementation of the DasContract editor and an environment to generate an executable blockchain smart contract was developed and published on GitHub [139]. Two complex case studies were created – a decentralized mortgage (Section 8.5) and tokenized EU elections (Section 8.6) as a validation of the proposed concept.

In the Chapter 9, all contributions of our research were summarized, evaluated, and the research questions answered. The main limitations of our result were also discussed. The application of the research results was discussed.

Finally, the Part VII presents our peer-reviewed publications with their citations and 22 student theses we supervised on topics related to our research. Other related materials, such as our coursebook DEMO for IT [A.12], are also mentioned.

## 10.2 Future Work

The author of the dissertation thesis suggests to further explore the following topics:

- Further experiments in process digitalization using DEMO – Conducting more experiments may reveal sufficient textual descriptions available at a necessary level of detail. In addition, quantifying the missing actor roles specification in the current experiment would be desirable, as mentioned in Section 6.4.2.2.

- Automation of the OER method – Conducting the OER method manually is elaborate. Although ontological analysis cannot be fully automated inherently, the promising research results in the area of NLP could help, such as [26, 47, 173]. Applying

such an approach to the first phase of the proposed OER method may reduce the need for manual processing.

- Extend the DasContract DSL language with a Decentralized Identity (DiD) [167] and Blockchain oracles such as ChainLink [147].
- Compare possibilities of executing the processes in public vs. private blockchains.
- Compare different blockchain implementations such as the Cardano [80], European Blockchain Services Infrastructure [41], and Hyperledger Fabric [153] and their ability to execute DasContract models.
- Evolvability of the proposed DasContract DSL with Normalized Systems Theory [98].

# Part VII

# Publications

# Bibliography


[1] Flexible manufacturing systems: an overview. In *Modeling, Simulation, and Control of Flexible Manufacturing Systems*, volume Volume 6 of *Series in Intelligent Control and Intelligent Automation*, pages 15–37. WORLD SCIENTIFIC, January 1999. URL: https://www.worldscientific.com/doi/abs/10.1142/9789812839763_0002, doi:10.1142/9789812839763_0002.

[2] Crypto Hackers Set for Record Year After Looting Over $3 Billion. *Bloomberg.com*, October 2022. URL: https://www.bloomberg.com/news/articles/2022-10-13/crypto-hackers-set-for-record-year-after-looting-over-3-billion.

[3] Sarbanes-Oxley Act. Sarbanes-oxley act. *Washington DC*, 2002. Publisher: Citeseer.

[4] Petr Ančinec. Open-source DEMO Construction and Process Model Designer, June 2019. URL: http://hdl.handle.net/10467/83222.

[5] Petr Ančinec. Domain-Specific Languages for Off-chain UI in Decentralized Applications, June 2021. URL: http://hdl.handle.net/10467/94542.

[6] Jos CM Baeten, Twan Basten, Twan Basten, and MA Reniers. *Process algebra: equational theories of communicating processes*, volume 50. Cambridge university press, 2010.

[7] Kakoli Bandyopadhyay, Peter P Mykytyn, and Kathleen Mykytyn. A framework for integrated risk management in information technology. *Management Decision*, 1999. Publisher: MCB UP Ltd.

[8] BIS. The Basel Framework, 2019. URL: https://www.bis.org/basel_framework/.

[9] Blockchair. Ethereum transactions per second, 2020. URL: https://blockchair.com/ethereum/charts/transactions-per-second?compare=bitcoin&interval=full.

[10] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. *Model-Driven Software Engineering in Practice: Second Edition*. Morgan & Claypool Publishers, 2nd edition, 2017.

[11] Michael Brenner. *Financial cryptography and data security : FC 2017 international workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, sliema, malta, april 7, 2017, revised selected papers*. Springer, Cham, Switzerland, 2017.

[12] Vitalik Buterin. Ethereum and oracles, 2014. Type: blog tex.referencetype: blog. URL: https://blog.ethereum.org/2014/07/22/ethereum-and-oracles/.

[13] Radek Buša. Designing WYSIWYG Web Forms, June 2018. URL: http://hdl.handle.net/10467/77488.

[14] Tomáš Bydžovský. A State Management in Multi-client Single Page Web Applications, June 2019. URL: http://hdl.handle.net/10467/83141.

[15] Tomáš Bydžovský. Decentralized Identity in DasContract Decentralized Applications, June 2021. URL: http://hdl.handle.net/10467/94500.

[16] Artur Caetano, Aurélio Assis, José Borbinha, and José Tribolet. An Application of the Psi-Theory to the Analysis of Business Process Models. *SpringerLink*, pages 258–267, 2013. doi:10.1007/978-3-642-36611-6_24.

[17] Camunda. Camunda BPM System, 2022. URL: https://camunda.com/.

[18] Capgemini. The new future of Compliance and BPM, 2017. URL: https://www.capgemini.com/wp-content/uploads/2017/07/The_New_Future_of_Compliance_and_BPM.pdf.

[19] Josep Carmona. *Business process management : 15th international conference, BPM 2017, barcelona, spain, september 10-15, 2017, proceedings*. Springer, Cham, Switzerland, 2017.

[20] Cary Coglianese and Jennifer Nash. Compliance Management Systems: Do They Make a Difference?, August 2020. URL: https://papers.ssrn.com/abstract=3598264.

[21] CoinGeco. CoinGecko yield farming survey 2020, 2020. URL: https://www.coingecko.com/buzz/yield-farming-survey-2020?locale=en.

[22] OGC Office of Government Commerce. *The official introduction to the ITIL service lifecycle*. IT infrastructure library. Stationery Office, 2007. URL: https://books.google.cz/books?id=9uLkMMqRKrYC.

[23] Steve Cook, Conrad Bock, Pete Rivett, Tom Rutt, Ed Seidewitz, Bran Selic, and Doug Tolbert. Unified Modeling Language (UML) Version 2.5.1. Standard, Object Management Group (OMG), December 2017. URL: https://www.omg.org/spec/UML/2.5.1.

[24] José Cordeiro, Joaquim Filipe, and Kecheng Liu. A UML profile for enterprise ontology. *Enterprise Systems and Technology*, page 7, 2008.

[25] Kevin Curran. E-voting on the blockchain. *The Journal of the British Blockchain Association*, 1(2):4451, 2018. Publisher: The British Blockchain Association.

[26] Paulius Danenas, Tomas Skersys, and Rimantas Butleris. Natural language processing-enhanced extraction of SBVR business vocabularies and business rules from UML use case diagrams. *Data & Knowledge Engineering*, 128:101822, July 2020. URL: https://www.sciencedirect.com/science/article/pii/S0169023X1930299X, doi:10.1016/j.datak.2020.101822.

[27] Chris Dannen. *Introducing ethereum and solidity*. O'Reilly Media, Inc, Brooklyn, New York, USA, 2017.

[28] Joop de Jong. Designing the Information Organization from Ontological Perspective. In Wil van der Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw, Clemens Szyperski, Antonia Albani, Jan L. G. Dietz, and Jan Verelst, editors, *Advances in Enterprise Engineering V*, volume 79, pages 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[29] Jan Dietz. *Enterprise ontology theory and methodology*. Springer, Berlin New York, 2006.

[30] Jan Dietz. The discipline of enterprise engineering, 2014. Published: [ONLINE] [accessed: 25. 4. 2014]. URL: https://www.alexandria.unisg.ch/export/DL/224477.pdf.

[31] Jan Dietz, Jan Hoogervorst, Antonia Albani, David Aveiro, Eduard Babkin, Joseph Barjis, Artur Caetano, Philip Huysmans, J. Iijima, Steven Kervel, Hans Mulder, Martin Op 't Land, Henderik Proper, Jorge Sanz, Linda Terlouw, José Tribolet, Jan Verelst, and Robert Winter. The discipline of Enterprise Engineering. *International Journal of Organisational Design and Engineering*, 3:86–114, 2013. doi:10.1504/IJODE.2013.053669.

[32] Jan L. G. Dietz and Hans B. F. Mulder. *Enterprise Ontology: A Human-Centric Approach to Understanding the Essence of Organisation*. The Enterprise Engineering Series. Springer International Publishing, 2020. URL: https://www.springer.com/gp/book/9783030388539, doi:10.1007/978-3-030-38854-6.

[33] Jan L.G. Dietz. *The Essence of Organization - an Introduction to Enterprise Engineering*. Sapio bv, 2012.

[34] Remco Dijkman and Pieter Van Gorp. BPMN 2.0 execution semantics formalized as graph rewrite rules. In Jan Mendling, Matthias Weidlich, and Mathias Weske, editors, *Business process modeling notation*, pages 16–30, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[35] Remco M Dijkman, Marlon Dumas, and Chun Ouyang. Formal semantics and analysis of BPMN process models using Petri nets. *Queensland University of Technology, Tech. Rep*, pages 1–30, 2007.

[36] Martin Drozdík. Open-Source Legal Process Designer in .NET Blazor, June 2020. URL: http://hdl.handle.net/10467/88271.

[37] Bruno Borlini Duarte, Andre Luiz de Castro Leal, Ricardo de Almeida Falbo, Giancarlo Guizzardi, Renata S. S. Guizzardi, and Vítor E. Silva Souza. Ontological foundations for software requirements with a focus on requirements at runtime. *Applied Ontology*, 13(2):73–105, May 2018. URL: http://doi.org/10.3233/AO-180197, doi:10.3233/AO-180197.

[38] Emmy Dudok, Sérgio Guerreiro, Eduard Babkin, Robert Pergl, and Steven J. H. van Kervel. Enterprise Operational Analysis Using DEMO and the Enterprise Operating System. In David Aveiro, Robert Pergl, and Michal Valenta, editors, *Advances in Enterprise Engineering IX*, number 211 in Lecture Notes in Business Information Processing, pages 3–18. Springer International Publishing, June 2015. 00000. URL: http://link.springer.com/chapter/10.1007/978-3-319-19297-0_1, doi:10.1007/978-3-319-19297-0_1.

[39] Ondrej Dvorak and Robert Pergl. Tackling rapid technology changes by applying enterprise engineering theories. *Science of Computer Programming*, 215:23, March 2022. doi:https://doi.org/10.1016/j.scico.2021.102747.

[40] Céline Décosse, Wolfgang A. Molnar, and Henderik A. Proper. What Does DEMO Do? A Qualitative Analysis about DEMO in Practice: Founders, Modellers and Beneficiaries. In Wil van der Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw, Clemens Szyperski, David Aveiro, José Tribolet, and Duarte Gouveia, editors, *Advances in Enterprise Engineering VIII*, volume 174, pages 16–30. Springer International Publishing, Cham, 2014.

[41] EBSI. Meet EUBlockchain: The European Blockchain Services Infrastructure (EBSI), 2019. URL: https://www.youtube.com/watch?v=mpbWQbkl8_g.

[42] S. Ellis, A. Juels, and S. Nazarov. ChainLink: A decentralized oracle network. 2017. URL: https://link.smartcontract.com/whitepaper.

[43] Ethereum. Ethereum project. URL: https://ethereum.org/.

[44] ethernodes.org. The ethereum nodes explorer. tex.howpublished: [online]. URL: https://www.ethernodes.org.

[45] European Commission. Shaping Europe's digital future: Blockchain Strategy, 2022. URL: https://digital-strategy.ec.europa.eu/en/policies/blockchain-strategy.

[46] European Parliament. General Data Protection Regulation (GDPR), 2016. URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[47] Shaokun Fan, Zhimin Hua, Veda C. Storey, and J. Leon Zhao. A process ontology based approach to easing semantic ambiguity in business process modeling. *Data & Knowledge Engineering*, 102:57–77, 2016. URL: https://www.sciencedirect.com/science/article/pii/S0169023X16000112, doi:https://doi.org/10.1016/j.datak.2016.01.001.

[48] Filip Fatz, Philip Hake, and Peter Fettke. Towards tax compliance by design: A decentralized validation of tax processes using blockchain technology. In *2019 IEEE 21st conference on business informatics (CBI)*, volume 01, pages 559–568, 2019. doi:10.1109/CBI.2019.00071.

[49] Carlos Figueira and David Aveiro. A New Action Rule Syntax for DEmo MOdels Based Automatic worKflow procEss geneRation (DEMOBAKER). In David Aveiro, José Tribolet, and Duarte Gouveia, editors, *Advances in Enterprise Engineering VIII*, number 174 in Lecture Notes in Business Information Processing, pages 46–60. Springer International Publishing, May 2014.

[50] ForMetis. Online DEMO modeling tool, 2011. Published: [ONLINE] [accessed: 25. 4. 2014]. URL: https://www.demoworld.nl/Portal/Home.

[51] Jan Frait. Generating Ethereum Smart Contracts from DasContract Language, August 2020. URL: http://hdl.handle.net/10467/90034.

[52] Luciano García-Bañuelos, Alexander Ponomarev, Marlon Dumas, and Ingo Weber. Optimized execution of business processes on blockchain. September 2017. doi:10.1007/978-3-319-65000-5_8.

[53] Bryan A. Garner. *Black's Law Dictionary: Deluxe Ninth Edition*. West, 2009.

[54] Gartner. The Reality of Blockchain, 2019. URL: https://www.gartner.com/smarterwithgartner/the-reality-of-blockchain/.

[55] Gartner. Gartner Information Technology Glossary, 2022. URL: https://www.gartner.com/en/information-technology/glossary.

[56] Github. Solidity for blockly. URL: https://github.com/promethe42/blockly-solidity.

[57] Google. Blockly. URL: https://developers.google.com/blockly/.

[58] Duarte Gouveia and David Aveiro. Modeling the system described by the EU General Data Protection Regulation with DEMO. In *Enterprise Engineering Working Conference*, pages 144–158. Springer, 2018.

[59] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[60] Giancarlo Guizzardi. *Ontological Foundations for Structural Conceptual Models*, volume 015. University of Twente, Enschede, 2005.

[61] Giancarlo Guizzardi, Luís Ferreira Pires, and Marten Sinderen. An Ontology-Based Approach for Evaluating the Domain Appropriateness and Comprehensibility Appropriateness of Modeling Languages. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Lionel Briand, and Clay Williams, editors, *Model Driven Engineering Languages and Systems*, volume 3713, pages 691–705. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. URL: http://80.link.springer.com.dialog.cvut.cz/chapter/10.1007/11557432_51.

[62] Giancarlo Guizzardi and Gerd Wagner. Can BPMN be used for making simulation models? *Lecture Notes in Business Information Processing*, 88 LNBIP:100–115, 2011.

[63] J. Habermas and T. McCarthy. *The Theory of Communicative Action: Reason and the rationalization of society*. The Theory of Communicative Action. Beacon Press, 1984. URL: https://books.google.cz/books?id=kuFhjNZuHTAC.

[64] Haes. *Enterprise governance of information technology : achieving alignment and value, featuring COBIT 5*. Springer, Cham, 2015.

[65] Sharon Halliday, Karin Badenhorst, and Rossouw Von Solms. A business approach to effective information technology risk analysis and management. *Information Management & Computer Security*, 1996. Publisher: MCB UP Ltd.

[66] Terry Halpin. Fact-Oriented Modeling: Past, Present and Future. In *Conceptual Modelling in Information Systems Engineering*, pages 19–38. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[67] Terry Halpin. *Object-Role Modeling Fundamentals: A Practical Guide to Data Modeling with ORM*. Technics Publications, Basking Ridge, NJ, first edition edition, April 2015.

[68] Michael Hammer. What is business process management? In Jan vom Brocke and Michael Rosemann, editors, *Handbook on business process management 1: Introduction, methods, and information systems*, pages 3–16. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. doi:10.1007/978-3-642-45100-3.

[69] Hands on Banking. Steps in the lending process. 2017. tex.howpublished: [online]. URL: https://handsonbanking.org/adults/buying-home/getting-mortgage/steps-in-the-lending-process/.

[70] Paul Webster Hare. *Making diplomacy work: intelligent innovation for the modern world*. CQ Press, 2015.

[71] Mustafa Hashmi, Guido Governatori, Ho-Pun Lam, and Moe Thandar Wynn. Are we done with business process compliance: state of the art and challenges ahead. *Knowledge and Information Systems*, 57(1):79–133, October 2018. doi:10.1007/s10115-017-1142-1.

[72] S. Heller. Usage of DEMO Methods for BPMN Models Creation. Master's thesis, Czech Technical University in Prague. Computing and Information Centre., 2016. URL: http://hdl.handle.net/10467/62776.

[73] Henrique Henriques, Hugo Lourenço, Vasco Amaral, and Miguel Goulão. Improving the Developer Experience with a Low-Code Process Modelling Language. In *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, MODELS '18, pages 200–210, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3239372.3239387.

[74] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design Science in Information Systems Research. *MIS Q.*, 28(1):75–105, March 2004.

[75] HHS. Summary of the HIPAA Security Rule, 2013. URL: https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html.

[76] Barbora Hornáčková. Using Blockchain Smart Contracts in the DEMO Methodology, January 2018. URL: http://hdl.handle.net/10467/74045.

[77] Barbora Hornáčková, Marek Skotnica, and Robert Pergl. Exploring a Role of Blockchain Smart Contracts in Enterprise Engineering. In David Aveiro, Giancarlo Guizzardi, Sérgio Guerreiro, and Wided Guédria, editors, *Advances in Enterprise Engineering XII*, pages 113–127, Cham, 2019. Springer International Publishing.

[78] F Hunka and S van Kervel. A generic DEMO model for co-creation and co-production as a basis for a truthful and appropriate REA model representation. *International Conference on Business Process ...*, 2019. Publisher: Springer. URL: https://link.springer.com/chapter/10.1007/978-3-030-30429-4_14, doi:10.1007/978-3-030-30429-4_14.

[79] Gregory Hutchins. *ISO 31000: 2018 enterprise risk management*. Greg Hutchins, 2018.

[80] IOHK. Cardano Blockchain, 2017. URL: https://why.cardano.org/en/introduction/motivation/.

[81] ISO. ISO/IEC 27000 family - Information security management systems, 2019. URL: https://www.iso.org/isoiec-27001-information-security.html.

[82] ISO. International Organization for Standardization (ISO), 2022. URL: https://www.iso.org.

[83] Barbora Jančovičová. Next Generation Methods for Development of Enterprise Information Systems, February 2019. URL: http://hdl.handle.net/10467/80244.

[84] Thomas Johanndeiter, Anat Goldstein, and Ulrich Frank. Towards Business Process Models at Runtime. *MoDELS@ Run. time*, 1079:13–25, 2013.

[85] Seema Kedar. *Programming Paradigms And Methodology.* Technical Publications, January 2008.

[86] Steven JH Van Kervel, John Hintzen, Tycho van Meeuwen, Joost Vermolen, and Bob Zijlstra. A professional case management system in production, modeled and implemented using DEMO. In Molnar, Wolfgang A., Henderik A. Proper, Jelena Zdravkovic, Peri Loucopoulos, Oscar Pastor, and Sybren de Kinderen, editors, *Complementary proceedings of the 8th Workshop on Transformation & Engineering of Enterprises (TEE 2014), and the 1st International Workshop on Capability-oriented Business Informatics (CoBI 2014)*, volume 1182, Geneva, Switzerland, July 2014. Technical University of Aachen.

[87] Meriem Kherbouche and Bálint Molnár. Formal Model Checking and Transformations of Models Represented in UML with Alloy. In Ajantha Dahanayake, Oscar Pastor, and Bernhard Thalheim, editors, *Modelling to Program*, Communications in Computer and Information Science, pages 127–136, Cham, 2021. Springer International Publishing. doi:10.1007/978-3-030-72696-6_6.

[88] KPMG. The future of Compliance, 2016. URL: https://assets.kpmg/content/dam/kpmg/ch/pdf/ch-ac-news-55-article-04-en.pdf.

[89] KPMG. Innovating compliance through automation, 2019. URL: https://assets.kpmg/content/dam/kpmg/bm/pdf/2019/02/ComplianceAutomationSurvey-bda-f-web.pdf.

[90] Katarina Krbilová. Process Mining in Finance Domain, January 2020. URL: http://hdl.handle.net/10467/86182.

[91] Marien R. Krouwel, Martin Opt Land, and Henderik A. Proper. Generating Low-Code Applications from Enterprise Ontology. In Balbir S. Barn and Kurt Sandkuhl, editors, *The Practice of Enterprise Modeling*, Lecture Notes in Business Information Processing, pages 18–32, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-031-21488-2_2.

[92] Vitus SW Lam. A precise execution semantics for BPMN. *IAENG International Journal of Computer Science*, 39(1):20–33, 2012.

[93] Pablo Lamela Seijas and Simon Thompson. Marlowe: Financial contracts on blockchain: 8th international symposium, ISoLA 2018, limassol, cyprus, november 5-9, 2018, proceedings, part IV. pages 356–375. November 2018. doi:10.1007/978-3-030-03427-6_27.

[94] Matěj Lang. WebAssembly Approach to Client-side Web Development using Blazor Framework, June 2019. URL: http://hdl.handle.net/10467/82332.

[95] Martina Lassaková. Law Modelling Using BPMN and DEMO, June 2019. URL: http://hdl.handle.net/10467/82328.

[96] Martina Lassaková, Marek Skotnica, and Robert Pergl. Czech Arbitration Court - Case Study Dataset, 2021. URL: https://github.com/CCMiResearch/DEMOCaseStudies/tree/master/ArbitrationCourt.

[97] Orlenys López-Pintado, Luciano García-Bañuelos, Marlon Dumas, Ingo Weber, and Alexander Ponomarev. CATERPILLAR: A Business Process Execution Engine on the Ethereum Blockchain. *CoRR*, abs/1808.03517, 2018. _eprint: 1808.03517. URL: http://arxiv.org/abs/1808.03517.

[98] H. Mannaert, P. De Bruyn, and J. Verelst. Exploring entropy in software systems: towards a precise definition and design rules. In *Proceedings of the Seventh International Conference on Systems (ICONS)*, pages 93–99, Saint Gilles, Reunion Island, 2012.

[99] Herwig Mannaert and Jan Verelst. *Normalized Systems—Re-creating Information Technology Based on Laws for Software Evolvability*. Koppa, Kermt, Belgium, 2009. 00049.

[100] Jan Mendling. Event-Driven Process Chains (EPC). In Jan Mendling, editor, *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*, Lecture Notes in Business Information Processing, pages 17–57. Springer, Berlin, Heidelberg, 2008. doi:10.1007/978-3-540-89224-3_2.

[101] MGIC for Homebuyers. The mortgage process. 2017. tex.howpublished: [online]. URL: https://homebuyers.mgic.com/getting-your-mortgage/mortgage-process.html.

[102] Stanislav Mikeš. Evolvability of Business Process Models, February 2019. URL: http://hdl.handle.net/10467/80236.

[103] Makiko Mita, Kensuke Ito, Shohei Ohsawa, and Hideyuki Tanaka. What is stablecoin?: A survey on price stabilization mechanisms for decentralized payment systems.

In *2019 8th international congress on advanced applied informatics (IIAI-AAI)*, pages 60–66, 2019. tex.organization: IEEE.

[104] Scott Mitchell. *GRC capability model version 2.1.* OCEG, Scottsdale, Ariz, 2012.

[105] Hamid R. Motahari-Nezhad and Keith D. Swenson. Adaptive Case Management: Overview and Research Challenges. In *2013 IEEE 15th Conference on Business Informatics*, pages 264–269, July 2013. ISSN: 2378-1971. doi:10.1109/CBI.2013.44.

[106] Ondřej Mráz, Pavel Náplava, Robert Pergl, and Marek Skotnica. Converting DEMO PSI Transaction Pattern into BPMN: A Complete Method. In David Aveiro, Robert Pergl, Giancarlo Guizzardi, João Paulo Almeida, Rodrigo Magalhães, and Hans Lekkerkerk, editors, *Advances in Enterprise Engineering XI: 7th Enterprise Engineering Working Conference, EEWC 2017, Antwerp, Belgium, May 8-12, 2017, Proceedings*, pages 85–98. Springer International Publishing, Cham, 2017. URL: http://dx.doi.org/10.1007/978-3-319-57955-9_7, doi:10.1007/978-3-319-57955-9_7.

[107] Martin Mužák. Model-Driven Approach to Governance, Risk, and Compliance Systems Development, June 2019. URL: http://hdl.handle.net/10467/82311.

[108] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. tex.added-at: 2014-04-17T08:33:06.000+0200 tex.biburl: https://www.bibsonomy.org/bibtex/23db66df0fc9fa2b5033f096a901f1c36/ngnn tex.interhash: 423c2cdff70ba0cd0bca55ebb164d770 tex.intrahash: 3db66df0fc9fa2b5033f096a901f1c36 tex.timestamp: 2014-04-17T08:33:06.000+0200. URL: http://www.bitcoin.org/bitcoin.pdf.

[109] NEO. NEO smart contract introduction. Type: web page. URL: http://docs.neo.org/en-us/sc/introduction.html.

[110] Blockchain News. DeFi's smart contract risks: Cream finance's input error led to CREAM token plunging 25%, 2020. URL: https://blockchain.news/news/defi-smart-contract-risks-cream-finance-input-error-token-plunge.

[111] Alex Norta. Designing a smart-contract application layer for transacting decentralized autonomous organizations. In *Advances in computing and data sciences: First international conference, ICACDS 2016, ghaziabad, india, november 11-12, 2016, revised selected papers*, pages 595–604, Singapore, 2017. Springer Singapore.

[112] Petr Nymsa. Mobile Enterprise Architecture Process Analytic Tool Based on the DEMO Methodology, June 2018. URL: http://hdl.handle.net/10467/77482.

[113] Pavel Náplava and Robert Pergl. Empirical Study of Applying the DEMO Method for Improving BPMN Process Models in Academic Environment. In *2015 IEEE 17th Conference on Business Informatics*, volume 2, pages 18–26, July 2015. doi:10.1109/CBI.2015.12.

[114] OMG. Business Process Model and Notation (BPMN), Version 2.0, January 2011. URL: http://www.omg.org/spec/BPMN/2.0.

[115] OMG. Unified Modeling Language, version 2.5, March 2015. URL: http://www.omg.org/spec/UML/2.5.

[116] OMG. Case Management Model and Notation (CMMN), Version 1.1, December 2016. URL: https://www.omg.org/spec/CMMN/1.1/.

[117] Gilles Oorts, Herwig Mannaert, Peter De Bruyn, and Ilke Franquet. On the Evolvable and Traceable Design of (Under)graduate Education Programs. In *Advances in Enterprise Engineering X*, Lecture Notes in Business Information Processing, pages 86–100. Springer, Cham, May 2016.

[118] Martin Op 't Land and Jan L. G. Dietz. Benefits of Enterprise Ontology in Governing Complex Enterprise Transformations. In *Advances in Enterprise Engineering VI*, volume 110, pages 77–92. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[119] John O'Connor. How Cardano can help development in Africa, 2018. Type: blog tex.referencetype: blog. URL: https://cardanofoundation.org/press/how-cardano-can-help-development-in-africa/.

[120] Berwin Leighton Paisner. The speed of business - Innovation, business growth and the impact of regulation, 2013. URL: http://vm1.blplaw.com/expert-legal-insights/speed-of-business.

[121] Susanne Patig, Vanessa Casanova-Brito, and Barbara Vögeli. IT Requirements of Business Process Management in Practice – An Empirical Study. In Richard Hull, Jan Mendling, and Stefan Tai, editors, *Business Process Management*, Lecture Notes in Computer Science, pages 13–28, Berlin, Heidelberg, 2010. Springer. doi:10.1007/978-3-642-15618-2_4.

[122] Mayur Patil, Vijay Pimplodkar, Anuja R Zade, Vinit Vibhute, and Ratnakar Ghadge. A survey on voting system techniques. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(1):114–117, 2013.

[123] PCI. PCI Security, 2004. URL: https://www.pcisecuritystandards.org/pci_security/.

[124] Marc Pilkington. Blockchain technology: Principles and applications. In F. Xavier Olleros and Majlinda Zhegu, editors, *Research handbook on digital transformations*, 2015. URL: https://ssrn.com/abstract=2662660.

[125] Orlenys Pintado. Caterpillar: A Blockchain-Based Business Process Management System. 2017.

[126] Kasireddy Preethi. ELI5: What do we mean by "blockchains are trustless"? *Medium*, March 2017. URL: https://medium.com/@preethikasireddy/eli5-what-do-we-mean-by-blockchains-are-trustless-aa420635d5f6.

[127] PRIAC. International Arbitration Court of the Czech Commodity Exchange, 2019. URL: http://www.priac.cz/en/.

[128] Wolfgang Reisig. Petri nets and algebraic specifications. *Theoretical Computer Science*, 80(1):1–34, March 1991. URL: https://www.sciencedirect.com/science/article/pii/030439759190203E, doi:10.1016/0304-3975(91)90203-E.

[129] Remix. Remix - solidity IDE. tex.howpublished: [online]. URL: https://remix.readthedocs.io/en/latest/.

[130] Olivier Rikken. 3 smart contract misconceptions. 2017. tex.howpublished: [online]. URL: https://www.coindesk.com/3-common-smart-contract-misconceptions-explored/.

[131] Risconnect. The 5 Biggest Challenges to Effective Compliance Management, 2021. URL: https://riskonnect.com/compliance/the-5-biggest-challenges-to-effective-compliance-management/.

[132] Bernd Rucker and Jakob Freund. *Real-Life BPMN (4th edition)*. Independently Published, September 2019.

[133] Fabian Schär. Decentralized Finance: On Blockchain- and Smart Contract-Based Financial Markets, April 2021. URL: https://papers.ssrn.com/abstract=3843844, doi:10.20955/r.103.153-74.

[134] Bela Shrimali and Hiren B. Patel. Blockchain state-of-the-art: architecture, use cases, consensus, challenges and opportunities. *Journal of King Saud University - Computer and Information Sciences*, 34(9):6793–6807, October 2022. URL: https://www.sciencedirect.com/science/article/pii/S131915782100207X, doi:10.1016/j.jksuci.2021.08.005.

[135] Bruce Silver. *BPMN Method and Style, 2nd Edition, with BPMN Implementer's Guide: A structured approach for business process modeling and implementation using BPMN 2.0*. Cody-Cassidy Press, October 2011.

[136] Mayank Singh. *Advances in computing and data sciences : first International Conference, ICACDS 2016, Ghaziabad, India, November 11-12, 2016, Revised selected papers*. Springer, Singapore, 2017.

[137] Marek Skotnica, Marta Aparício, Robert Pergl, and Sérgio Guerreiro. Process digitalization using blockchain: EU parliament elections case study. In *Proceedings of the 9th international conference on model-driven engineering and software*

*development*. SCITEPRESS - Science and Technology Publications, 2021. doi:10.5220/0010229000650075.

[138] Marek Skotnica and et. al. DasContract 1.0 github repository, 2020. URL: https://github.com/CCMiResearch/DasContract/tree/v1.0.

[139] Marek Skotnica and et. al. DasContract github repository, 2021. URL: https://github.com/CCMiResearch/DasContract.

[140] Marek Skotnica, Steven J. H. van Kervel, and Robert Pergl. Towards the Ontological Foundations for the Software Executable DEMO Action and Fact Models. In *Advances in Enterprise Engineering X*, pages 151–165, Funchal, Madeira, May 2016. Springer International Publishing. URL: http://link.springer.com/chapter/10.1007/978-3-319-39567-8_10, doi:10.1007/978-3-319-39567-8_10.

[141] Marek Skotnica, Jan Klicpera, and Robert Pergl. Towards model-driven smart contract systems - code generation and improving expressivity of smart contract modeling. CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2825/paper1.pdf.

[142] Marek Skotnica and Robert Pergl. Das Contract - A Visual Domain Specific Language for Modeling Blockchain Smart Contracts. In David Aveiro, Giancarlo Guizzardi, and José Borbinha, editors, *Advances in Enterprise Engineering XIII*, pages 149–166, Cham, 2020. Springer International Publishing.

[143] Marek Skotnica, Robert Pergl, Lucie Havrdová, Viktor Holý, Michaela Kučerová, Jana Martínková, Daniel Matoušek, Jan Novotný, David Primus, Filip Sikora, Roman Soběslav, Jan Starůstka, Jan Stejskal, Ladislav Strnad, Albert Švehla, and Ondřej Cihlář. Czech Procedural Law Modeling - Case Study Dataset, 2022. URL: https://github.com/CCMiResearch/DEMOCaseStudies/tree/master/MEP/2021.

[144] Marek Skotnica, Robert Pergl, Anna Vitmanová, Jiří Růžička, Denis Drda, Katarina Krbilová, Lenka Obermajerová, Radka Bodnárová, Jan Bittner, Martin Drozdík, Jan Klicpera, Matyáš Herman, Ema Holínská, Jiří Zikán, Jan Horyna, Ondřej Šelder, Jiří Kasl, and Petr Kučera. Code of Civil Procedure - Case Study Dataset, 2021. URL: https://github.com/CCMiResearch/DEMOCaseStudies/tree/master/MEP/2020.

[145] Marek Skotnica, Steven J. H. van Kervel, and Robert Pergl. A DEMO Machine - A Formal Foundation for Execution of DEMO Models. In David Aveiro, Robert Pergl, Giancarlo Guizzardi, Joao Paulo Almeida, Rodrigo Magalhaes, and Hans Lekkerkerk, editors, *Advances in Enterprise Engineering XI: 7th Enterprise Engineering Working Conference, EEWC 2017, Antwerp, Belgium, May 8-12, 2017, Proceedings*, pages 18–32. Springer International Publishing, Cham, 2017. URL: http://dx.doi.org/10.1007/978-3-319-57955-9_2, doi:10.1007/978-3-319-57955-9_2.

[146] Smart Contracts Alliance. Smart contracts: 12 use cases for business & beyond. 2016. tex.howpublished: [online]. URL: http://digitalchamber.org/assets/smart-contracts-12-use-cases-for-business-and-beyond.pdf.

[147] Steve Ellis, et a. ChainLink - A Decentralized Oracle Network, September 2017. URL: https://link.smartcontract.com/whitepaper.

[148] Tetsuya Suga and Junichi Iijima. Algebra for Enterprise Ontology: towards analysis and synthesis of enterprise models. *Enterprise Information Systems*, 12(3):341–370, March 2018. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/17517575.2017.1367962. doi:10.1080/17517575.2017.1367962.

[149] Melanie Swan. *Blockchain : blueprint for a new economy*. O'Reilly, Sebastopol, Calif, 2015.

[150] Tim Swanson. Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems. 2016. tex.howpublished: [online]. URL: http://www.ofnumbers.com/wp-content/uploads/2015/04/Permissioned-distributed-ledgers.pdf.

[151] Norris Syed Abdullah, Shazia Sadiq, and Marta Indulska. Emerging Challenges in Information Systems Research for Regulatory Compliance Management. In Barbara Pernici, editor, *Advanced Information Systems Engineering*, Lecture Notes in Computer Science, pages 251–265, Berlin, Heidelberg, 2010. Springer. doi:10.1007/978-3-642-13094-6_21.

[152] Nick Szabo. Smart contracts: Building blocks for digital markets. *www.fon.hum.uva.nl*, 1996. URL: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html.

[153] The Hyperledger White Paper Working Group. An Introduction to Hyperledger, 2018. URL: https://www.hyperledger.org/wp-content/uploads/2018/08/HL_Whitepaper_IntroductiontoHyperledger.pdf.

[154] Weng Jie Thong and M. A. Ameedeen. A Survey of Petri Net Tools. In Hamzah Asyrani Sulaiman, Mohd Azlishah Othman, Mohd Fairuz Iskandar Othman, Yahaya Abd Rahim, and Naim Che Pee, editors, *Advanced Computer and Communication Engineering Technology*, Lecture Notes in Electrical Engineering, pages 537–551, Cham, 2015. Springer International Publishing. doi:10.1007/978-3-319-07674-4_51.

[155] Stepan Tužil. Automated Transformation of DEMO Models into BPMN, June 2019. URL: http://hdl.handle.net/10467/83117.

[156] Simon Urbánek. Exploring the use of blockchain smart contract in the e-commerce, June 2021. URL: http://hdl.handle.net/10467/94561.

[157] Vijay Vaishnavi and B Kuechler. Design Science Research in Information Systems. *Association for Information Systems*, January 2004.

[158] W. M. P. van der Aalst. Formalization and verification of event-driven process chains. *Information and Software Technology*, 41(10):639–650, July 1999. URL: https://www.sciencedirect.com/science/article/pii/S0950584999000166, doi:10.1016/S0950-5849(99)00016-6.

[159] W. M. P. van der Aalst and A. H. M. ter Hofstede. YAWL: yet another workflow language. *Information Systems*, 30(4):245–275, June 2005. URL: https://www.sciencedirect.com/science/article/pii/S0306437904000304, doi:10.1016/j.is.2004.02.002.

[160] Wil MP Van der Aalst. Business process management: a comprehensive survey. *International Scholarly Research Notices*, 2013, 2013. Publisher: Hindawi.

[161] Alta van der Merwe, Aurona Gerber, and Hanlie Smuts. Mapping a design science research cycle to the postgraduate research report. In Janet Liebenberg and Stefan Gruner, editors, *ICT education*, pages 293–308, Cham, 2017. Springer International Publishing.

[162] Alta van der Merwe, Aurona Gerber, and Hanlie Smuts. Guidelines for conducting design science research in information systems. In Bobby Tait, Jan Kroeze, and Stefan Gruner, editors, *ICT education*, pages 163–178, Cham, 2020. Springer International Publishing.

[163] W. Van Grembergen. Introduction to the minitrack "IT governance and its mechanisms" HICSS 2002. In *Proceedings of the 35th annual hawaii international conference on system sciences*, pages 3097–3097, 2002. doi:10.1109/HICSS.2002.994349.

[164] Steven J. H. Van Kervel, Jan L. G. Dietz, John Hintzen, Tycho van Meeuwen, and Bob Zijlstra. Enterprise Ontology Driven Software Engineering. In Slimane Hammoudi, Marten van Sinderen, and José Cordeiro, editors, *ICSOFT*, pages 205–210. SciTePress, 2012. URL: http://dblp.uni-trier.de/db/conf/icsoft/icsoft2012.html.

[165] Dieter Van Nuffel, Hans Mulder, and Steven Van Kervel. Enhancing the Formal Foundations of BPMN by Enterprise Ontology. In Will van der Aalst, John Mylopoulos, Norman M. Sadeh, Michael J. Shaw, Clemens Szyperski, Antonia Albani, Joseph Barjis, and Jan L. G. Dietz, editors, *Advances in Enterprise Engineering III*, volume 34, pages 115–129. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[166] Mark Von Rosing, Henrik Von Scheel, and August-Wilhelm Scheer. *The complete business process handbook: Body of knowledge from process modeling to BPM, volume 1*, volume 1. Morgan Kaufmann, 2014.

[167] W3C. Decentralized Identifiers (DIDs) v1.0, September 2020. URL: https://www.w3.org/TR/did-core/.

[168] 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.

[169] Roel Wieringa. Design science methodology: principles and practice. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*, pages 493–494. ACM, 2010.

[170] YChart. Ethereum average transaction fee, 2020. URL: `https://ycharts.com/indicators/ethereum_average_transaction_fee`.

[171] Peng Zhang, Douglas C. Schmidt, Jules White, and Gunther Lenz. Chapter One - Blockchain Technology Use Cases in Healthcare. In Pethuru Raj and Ganesh Chandra Deka, editors, *Advances in Computers*, volume 111 of *Blockchain Technology: Platforms, Tools and Use Cases*, pages 1–41. Elsevier, January 2018. URL: `https://www.sciencedirect.com/science/article/pii/S0065245818300196`, `doi:10.1016/bs.adcom.2018.03.006`.

[172] Ondřej Šelder. Generating Plutus Smart Contracts from DEMO Process Models, January 2020. URL: `http://hdl.handle.net/10467/86189`.

[173] David Šenkýř, Marek Suchánek, Petr Kroha, Herwig Mannaert, and Robert Pergl. Expanding Normalized Systems from textual domain descriptions using TEMOS. *Journal of Intelligent Information Systems*, pages 1–24, 2022. Publisher: Springer.

# Reviewed Publications of the Author Relevant to the Thesis

[A.1] Skotnica M. (70%); van Kervel S.J.H.(10%); Pergl R.(20%) Towards the Ontological Foundations for the Software Executable DEMO Action and Fact Models In: *Advances in Enterprise Engineering X. EEWC 2016. Lecture Notes in Business Information Processing.* Springer, Cham, 2016.

The paper has been cited in:

- Huňka F.; van Kervel S.J.H. The REA Model Expressed in a Generic DEMO Model for Co-creation and Co-production. In: *Advances in Enterprise Engineering XI. EEWC 2017. Lecture Notes in Business Information Processing.* Springer, Cham, 2017.
- Mouhib N.; Bah S.; Berrado A. The viable system model driven the organization and the information system design. In: *2018 International Conference on Intelligent Systems and Computer Vision (ISCV).* IEEE, 2018.
- Huňka F.; van Kervel S.J.H.; Matula J. The DEMO Co-creation and Co-production Model and Its Utilization. In: *Enterprise and Organizational Modeling and Simulation. EOMAS 2018. Lecture Notes in Business Information Processing.* Springer, Cham, 2018.
- Huňka F.; van Kervel S.J.H.; Contract Modeling Utilizing DEMO Co-creation Coproduction Model. In: *12th International Workshop on Value Modeling and Business Ontologies, VMBO 2018.* Amsterdam, 2018
- Huňka F.; van Kervel S.J.H.; A Generic DEMO Model for Co-creation and Co-production as a Basis for a Truthful and Appropriate REA Model Representation. In: *Business Process Management: Blockchain and Central and Eastern Europe Forum. BPM 2019. Lecture Notes in Business Information Processing.* Springer, Cham, 2019.

- Mouhib N.; Bah S.; Berrado A. The Viable System Ontology Theory. In: *4th World Conference on Complex Systems (WCCS).* IEEE, 2019.
- Mouhib N.; Bah S.; Berrado A. Viability Theory and PSI Theory Interrelation Inspired by Bunge Systemic Classification: the Viable System Ontology Theory. In: *Systemic Practice and Action Research volume 33.* Springer, Cham, 2020.

[A.2] Mráz O. (40%); Náplava P. (20%); Pergl R. (20%); Skotnica M. (20%) Converting DEMO PSI Transaction Pattern into BPMN: A Complete Method. In: *Advances in Enterprise Engineering XI. EEWC 2017. Lecture Notes in Business Information Processing.* Springer, Cham, 2017.

The paper has been cited in:

- Babkin E.; Malyzhenkov P.; Yavorskiy C. Towards Model-Driven Role Engineering in BPM Software Systems. In: *Information Systems: Research; Development; Applications; Education. SIGSAND/PLAIS 2019. Lecture Notes in Business Information Processing.* Springer; Cham; 2019.
- Gray T.; Bork D.; De Vries M. A New DEMO Modelling Tool that Facilitates Model Transformations. In: *Enterprise, Business-Process and Information Systems Modeling. BPMDS 2020, EMMSAD 2020. Lecture Notes in Business Information Processing.* Springer, Cham, 2020.
- Mulder M.A.T.; Proper H.A. Towards Enterprise-Grade Tool Support for DEMO. In: *The Practice of Enterprise Modeling. PoEM 2020. Lecture Notes in Business Information Processing.* Springer, Cham, 2020.
- Gray T.; De Vries M. Empirical Evaluation of a New DEMO Modelling Tool that Facilitates Model Transformations. In: *Advances in Conceptual Modeling. ER 2020. Lecture Notes in Computer Science.* Springer, Cham, 2020.
- De Vries M.; Bork D. Identifying Scenarios to Guide Transformations from DEMO to BPMN. In: *Advances in Enterprise Engineering XIV. EEWC 2020. Lecture Notes in Business Information Processing.*, Springer, Cham, 2021.
- Mulder M.A.T.; Proper H.A. On the Development of Enterprise-Grade Tool Support for the DEMO Method. In: *Advanced Information Systems Engineering. CAiSE 2021. Lecture Notes in Computer Science.* Springer, Cham, 2021.
- Mulder M.A.T.; Proper H.A. On Enterprise-Grade Tool Support for DEMO. *Software and Systems Modeling (2021).* Springer, Cham, 2021.
- Krouwel M.R.; Land M.O.'.; Proper H.A. Generating Low-Code Applications from Enterprise Ontology. In: *The Practice of Enterprise Modeling. PoEM 2022. Lecture Notes in Business Information Processing, vol 456.* Springer, Cham. 2022.
- Mulder M.A.T.; Proper H.A. On Enterprise-Grade Tool Support for DEMO. In: *On Enterprise-Grade Tool Support for DEMO. Softw Syst Model 21, 1341–1361.* Springer, Cham. 2022.

[A.3] Skotnica M. (80%); van Kervel S.J.H. (5%); Pergl R. (15%) A DEMO Machine - A Formal Foundation for Execution of DEMO Models. In: *Advances in Enterprise Engineering XI. EEWC 2017. Lecture Notes in Business Information Processing.* Springer, Cham, 2017.

The paper has been cited in:

- Gouveia D.; Aveiro D. Colored Petri-Net for Implementing DEMO/PSI Transactions for N Actor Roles ($N >= 2$). In: *Advances in Enterprise Engineering XII. EEWC 2018. Lecture Notes in Business Information Processing.* Springer, Cham, 2019.
- Aparício M.; Guerreiro S.; Sousa P. Automated DEMO Action Model Implementation using Blockchain Smart Contracts. In: *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2020).* SciTePress, 2020.

[A.4] Hornáčková B. (20%); Skotnica M. (60%); Pergl R. (20%) Exploring a Role of Blockchain Smart Contracts in Enterprise Engineering. In: *Advances in Enterprise Engineering XII. EEWC 2018. Lecture Notes in Business Information Processing.* Springer, Cham, 2019.

The paper has been cited in:

- Nanayakkara S.; Rodrigo M.N.N.; Perera S.; Weerasuriya G.T. ; Hijazi A. A. A methodology for selection of a Blockchain platform to develop an enterprise system. In: *Journal of Industrial Information Integration Volume 23.* Elsevier, 2021.
- Lauster C.; Klinger P.; Schwab N.; Bodendorf F. Literature Review Linking Blockchain and Business Process Management. In: *15th International Conference on Wirtschaftsinformatik.* Germany, 2020
- Aparício, M.; Guerreiro S.; Sousa P. Towards an Automated DEMO Action Model Implementation using Blockchain Smart Contracts In: *Proceedings of the 22nd International Conference on Enterprise Information Systems (ICEIS 2020)* SciTePress, 2020.
- Aparício M.; Guerreiro S.; Sousa P. Automated DEMO Action Model Implementation using Blockchain Smart Contracts. In: *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2020)* SciTePress, 2020.
- Babkin E.; Komleva N. Model-Driven Liaison of Organization Modeling Approaches and Blockchain Platforms. In: *Advances in Enterprise Engineering XIII. EEWC 2019. Lecture Notes in Business Information Processing.* Springer, Cham, 2020.

- Levasseur O.; Iqbal M.; Matulevičius R. Survey of Model-Driven Engineering Techniques for Blockchain-Based Applications. In: *PoEM'21 Forum: 14th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modelling.* CEUR Workshop Proceedings (CEUR-WS.org), 2021.
- Nanayakkara S.; Rodrigo M.N.N.; Perera S.; Weerasuriya G.T.; Hijazi A.A. A methodology for selection of a Blockchain platform to develop an enterprise system. In: *Journal of Industrial Information Integration Volume 23.* Elsevier, 2021.

[A.5] Skotnica M. (80%); Pergl R. (20%) Das Contract - A Visual Domain Specific Language for Modeling Blockchain Smart Contracts. In: *Advances in Enterprise Engineering XIII. EEWC 2019. Lecture Notes in Business Information Processing.* Springer, Cham, 2020.

The paper has been cited in:

- Gómez O.; Rosero R.; Cortés-Verdín,K. CRUDyLeaf: A DSL for Generating Spring Boot REST APIs from Entity CRUD Operations. In: *Cybernetics and Information Technologies.* Sciendo, Bulgaria, 2020.
- Hsain Y. A.; Laaz N.; Mbarki S. Ethereum's Smart Contracts Construction and Development using Model Driven Engineering Technologies: a Review. In: *Procedia Computer Science.* Elsevier, 2021.
- Vieira M. L. L.; Vilain P. Representation of Smart Contracts as State Diagrams. In: *IEEE/ACS 18th International Conference on Computer Systems and Applications (AICCSA).* IEEE, 2021.
- Alam M. T.; Chowdhury S.; Halder R.; Maiti A. Blockchain Domain-Specific Languages: Survey, Classification, and Comparison. In: *IEEE International Conference on Blockchain (Blockchain).* IEEE, 2021.
- Merlec M.M.; Lee Y.K.; In H.P. SmartBuilder: A Block-based Visual Programming Framework for Smart Contract Development. In: *IEEE International Conference on Blockchain (Blockchain).* IEEE, 2021.
- Hsain Y.A.; Laaz N.; Mbarki S. Ethereum's Smart Contracts Construction and Development using Model Driven Engineering Technologies: a Review. In: *Procedia Computer Science.* Elsevier, 2021.
- Meyer E.; Welpe I. M.; Sandner P.G.; Decentralized Finance—A Systematic Literature Review and Research Directions. In: *ECIS 2022 Research Papers.* SSRN, 2022.
- Curty S.; Härer F.; Fill HG. Blockchain Application Development Using Model-Driven Engineering and Low-Code Platforms: A Survey. In: *BPMDS EMMSAD 2022 2022. Lecture Notes in Business Information Processing, vol 450.* Springer, Cham, 2022.

- Hunn P.; Allen J. Smart Legal Contracts: Computable Law in Theory and Practice. In: *Book* Oxford University Press, 2022.
- Dixit A.; Deval V.; Dwivedi V.; Norta A.; Draheim D. Towards user-centered and legally relevant smart-contract development: A systematic literature review. In: *Journal of Industrial Information Integration 26.* Elsevier, 2022.
- M.; Bernardi S.; Marrone S.; Merseguer J. An approach for the automatic verification of blockchain protocols: the Tweetchain case study. In: *Journal of Computer Virology and Hacking Techniques.* Elsevier, 2022.
- Hamdaqa M.; Met L.A.P.; Qasse I. iContractML 2.0: A domain-specific language for modeling and deploying smart contracts onto multiple blockchain platforms. In: *Information and Software Technology 144.* Elsevier, 2022.

[A.6] Skotnica M. (50%); Klicpera J. (40%); Pergl R. (10%) Towards model-driven smart contract systems - code generation and improving expressivity of smart contract modeling In: *EEWC Forum 2020.* CEUR-WS.org, 2021.

The paper has been cited in:

- van den Heuvel WJ.; Tamburri D.A.; D'Amici D.; Izzo F.; Potten S. ChainOps for Smart Contract-Based Distributed Applications. In: *Business Modeling and Software Design. BMSD 2021. Lecture Notes in Business Information Processing.* Springer, Cham, 2021.
- Samreen N.F.; Secure MDE for Ethereum-based Decentralized Applications (DApps) Development. In: *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C).* IEEE, 2021.
- Bistarelli S.; Faloci F.; Mori P. .chain: automatic coding of smart contracts and user interfaces for supply chains. In: *Third International Conference on Blockchain Computing and Applications (BCCA).* IEEE, 2021.
- Sousa V.A.; Burnay C. MDE4BBIS: A Framework to Incorporate Model-Driven Engineering in the Development of Blockchain-Based Information Systems. In: *Third International Conference on Blockchain Computing and Applications (BCCA).* IEEE, 2021.
- Meyer E.; Welpe I. M.; Sandner P.G.; Decentralized Finance—A Systematic Literature Review and Research Directions. In: *ECIS 2022 Research Papers.* SSRN, 2022.
- Baresi L.; Quattrocchi G.; Tamburri D.A.; Terracciano L. A declarative modelling framework for the deployment and management of blockchain applications. In: *MODELS '22: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems.* ACM, 2022.

- Jurgelaitis M.; Čeponienė L.; Butkus K.; Butkienė R.; Drungilas V. MDA-Based Approach for Blockchain Smart Contract Development. In: *Advances in Information System Analysis and Modeling (AISAM).* MDPI, 2022.
- Bistarelli S.; Faloci F.; Mori P. Towards a Graphical DSL for Tracing Supply Chains on Blockchain. In: *Euro-Par 2021: Parallel Processing Workshops.* Springer, Cham, 2022.
- Aidin Rasti A.; Amyot D.; Parvizimosaed A.; Roveri M.; Logrippo L.; Anda A.A.; Mylopoulos J. Symboleo2SC: from legal contract specifications to smart contracts. In: *MODELS '22: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems.* ACM, 2022.
- Jurgelaitis M.; Čeponienė L.; Butkienė R. Solidity Code Generation From UML State Machines in Model-Driven Smart Contract Development. In: *IEEE Access (Volume: 10).* IEEE, 2022.

[A.7] Skotnica M. (50%); Aparício M. (30%); Pergl R. (10%); Guerreiro S. (10%) Process digitalization using blockchain: EU parliament elections case study. In: *Proceedings of the 9th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2021).* SciTePress, 2021.

The paper has been cited in:

- Alam M. T.; Chowdhury S.; Halder R.; Maiti A. Blockchain Domain-Specific Languages: Survey, Classification, and Comparison. In: *IEEE International Conference on Blockchain (Blockchain).* IEEE, 2021.
- Benabdallah A.; Audras A.; Coudert L.; Madhoun N.E.; Badra M. Analysis of Blockchain Solutions for E-Voting: A Systematic Literature Review. In: *IEEE Access (Volume: 10).* IEEE, 2022.

# Remaining Publications of the Author Relevant to the Thesis

[A.8] Skotnica M. Implementation of a module supporting the AM model in the Formetis DEMO Processor. Bachelor thesis. Czech Technical University in Prague, Faculty of Information Technology, 2014.

[A.9] Skotnica M. Towards the Foundations of Fact and Rules Ontology for Discrete Systems Master thesis. Czech Technical University in Prague, Faculty of Information Technology, 2016.

[A.10] Skotnica M. Business Process Management Systems Based on Enterprise Engineering Discipline. In: *Doctoral Consortium Workshop, Enterprise Engineering Working Conference (EEWC 2017).* Antwerp, Belgium, 2017.

[A.11] Skotnica M. Design of Systems Supporting Human Cooperation and Co-production. Ph.D. Minimum Thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.

[A.12] Skotnica M. DEMO for IT A Textbook for MI-MEP - Modeling of Enterprise Processes. Czech Technical University in Prague, Faculty of Information Technology, 2018.

# Selected Relevant Supervised Theses

[A.13] Buša R. Designing WYSIWYG Web Forms. Bachelor thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.

[A.14] Nymsa P. Mobile Enterprise Architecture Process Analytic Tool Based on the DEMO Methodology. Bachelor thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.

[A.15] Ančinec P. Open-source DEMO Construction and Process Model Designer. Bachelor thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

[A.16] Bydžovský T. A State Management in Multi-client Single Page Web Applications. Bachelor thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

The bachelor thesis has been awarded:

- Dean's award for the best bachelor thesis of summer semester in the school year 2018/2019 on Czech Technical University in Prague, Faculty of Information Technology

[A.17] Drozdík M. Open-Source Legal Process Designer in .NET Blazor. Bachelor thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

[A.18] Krbilová K. Process Mining in Finance Domain. Bachelor thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

[A.19] Šelder O. Generating Plutus Smart Contracts from DEMO Process Models. Bachelor thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

[A.20] Škrabal M. Use Cases for Decentralized Identity. Bachelor thesis. Czech Technical University in Prague, Faculty of Information Technology, 2022.

[A.21] Hornáčková B. Using Blockchain Smart Contracts in the DEMO Methodology. Master thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.

The master thesis has been awarded:

- Dean's award for the best master thesis of summer semester in the school year 2017/2018 on Czech Technical University in Prague, Faculty of Information Technology

[A.22] Jančovičová B. Next Generation Methods for Development of Enterprise Information Systems. Master thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

[A.23] Lang M. WebAssembly Approach to Client-side Web Development using Blazor Framework. Master thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

[A.24] Lassaková M. Law Modelling Using BPMN and DEMO. Master thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

[A.25] Mikeš S. Evolvability of Business Process Models. Master thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

[A.26] Mužák M. Model-Driven Approach to Governance, Risk, and Compliance Systems Development. Master thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

[A.27] Frait J. Generating Ethereum Smart Contracts from DasContract Language. Master thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

[A.28] Bydžovský T. Decentralized Identity in DasContract Decentralized Applications. Master thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

[A.29] Ančinec P. Domain-Specific Languages for Off-chain UI in Decentralized Applications. Master thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

[A.30] Urbánek Š. Exploring the use of Blockchain Smart Contract in the E-Commerce. Master thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

[A.31] Drozdík M. Generation of Plutus Smart Contracts from DasContract models. Master thesis. Czech Technical University in Prague, Faculty of Information Technology, 2022.

[A.32] Klicpera J. Client-Side Application Development Using Blazor Framework - a Blockchain Smart Contract Designer Case Study. Master thesis. Czech Technical University in Prague, Faculty of Information Technology, 2022.

The master thesis has been awarded:

- Dean's award for the best master thesis of summer semester in the school year 2021/2022 on Czech Technical University in Prague, Faculty of Information Technology

[A.33] Krbilová K. Blockchain Smart Contracts in Public Sector. Master thesis. Czech Technical University in Prague, Faculty of Information Technology, 2022.

[A.34] Šelder O. Business Rules in Blockchain Smart Contracts. Master thesis. Czech Technical University in Prague, Faculty of Information Technology, 2022.