Assignment of master's thesis

Title:	Digitalization of Tax Administration Processes Using BPM Systems and DEMO Methodology
Student:	Bc. Daniel Matoušek
Supervisor:	Ing. Marek Skotnica
Study program:	Informatics
Branch / specialization:	Software Engineering
Department:	Department of Software Engineering
Validity:	until the end of summer semester 2023/2024

Instructions

The law consists of two main parts – substantive and procedural. The substantive law declares the rights and obligations of people, companies, and governments. The procedural law defines a process to follow when complying with the laws. The legal processes are notoriously complex and hard to digitize. The main goal of this thesis is to explore how techniques researched and taught at the Czech Technical University, such as DEMO and BPMN, can be applied to digitizing tax administration processes.
- Review state-of-the-art BPM systems, DEMO methodology, and BPMN.
- Create an as-is model of selected tax administration processes.
- Create a to-be digitalization model using BPMN.
- Create a proof of concept implementation in the Camunda BPM system.

Electronically approved by Ing. Michal Valenta, Ph.D. on 15 November 2022 in Prague.

Master's thesis

DIGITALIZATION OF TAX ADMINISTRATION PROCESSES USING BPM SYSTEMS AND DEMO METHODOLOGY

Bc. Daniel Matoušek

Faculty of Information Technology
Department of Software Engineering
Supervisor: Ing. Marek Skotnica
April 24, 2023

Contents

List of Figures

List of Tables

List of Code listings

I would like to express my deep gratitude to my thesis supervisor, Ing. Marek Skotnica, for his professional guidance, encouragement, and valuable advice throughout the creation of this thesis. I am also grateful to my family for their unwavering support and understanding during this academic journey. Their love, patience, and belief in me have been a constant source of strength and motivation.

Declaration

Abstract

The digitalization of legal processes can create a more streamlined and effective approach to legal proceedings, resulting in improved transparency, faster decision-making, better traceability, and an overall expedited process execution. However, since legal processes are usually subject to a complex web of regulations and strictly defined procedures, digitizing processes that are declared in legislation is much more challenging compared to digitizing business processes. This thesis focuses on the possible simplification of the development process of systems that support legal processes through the involvement of Design and Engineering Methodology for Organizations (DEMO) and Business Process Model and Notation (BPMN), researched at the Czech Technical University in Prague. The goal of the thesis is to create a proof-of-concept Business Process Management (BPM) system supporting the tax administration process defined in the Czech code of law. To realize this system, the DEMO methodology is used to analyse the as-is state of the tax administration process, while BPMN captures the to-be state of the process after digitization, providing the basis for implementation. The result of the thesis is a proof-of-concept BPM system based on the Camunda Platform 8 that supports the tax administration process.

Keywords Law digitalization, BPM system, DEMO, BPMN, Camunda Platform 8

Abstrakt

Digitalizace právních procesů může vytvořit efektivnější a přehlednější přístup k vykonávání právních řízení, což by vedlo ke zlepšení transparentnosti, rychlejším rozhodnutím, lepší sledovatelnosti a celkově urychlenému průběhu procesů. Avšak digitální zpracování procesů, které jsou deklarovány v právních předpisech, je mnohem náročnější v porovnání s digitalizací business procesů, jelikož právní procesy jsou obvykle založeny na komplexních předpisech a přísně definovaných postupech. Tato práce se zaměřuje na možná zjednodušení procesu vývoje systémů podporujících právní procesy prostřednictvím zapojení Design and Engineering Methodology for Organizations (DEMO) a Business Process Model and Notation (BPMN), jež jsou cílem výzkumné činnosti na Českém vysokém učení technickém v Praze. Cílem práce je vytvořit proof-of-concept Business Process Management (BPM) systém podporující proces daňového řízení definovaný v českém zákoníku. K dosažení tohoto cíle je použita DEMO metodologie pro analýzu současného stavu daňového řízení, zatímco BPMN zachycuje požadovaný stav procesu po digitalizaci a poskytuje základ pro implementaci. Výsledkem práce je proof-of-concept BPM systém založený na Camunda Platform 8, jež podporuje proces daňového řízení.

Klíčová slova Digitalizace zákonů, BPM systém, DEMO, BPMN, Camunda Platform 8

Acronyms

AM	Action Model
ARM	Action Rule Specification
BPM	Business Process Management
BPMI	Business Process Management Initiation
BPMN	Business Process Model and Notation
CM	Cooperation Model
CSD	Coordination Structure Diagram
DEMO	Design and Engineering Methodology for Organisations
FEEL	Friendly Enough Expression Language
FM	Fact Model
JVM	Java Virtual Machine
OER	Organisation essence revealing
OFD	Object Fact Diagram
OMG	Object Management Group
PM	Process Model
PSD	Process Structure Diagram
SaaS	Software as a service
TPT	Transaction Product Table
UML	Unified Modeling Language
WIS	Work Instruction Specification

Introduction

Business processes, from purchasing to production, and even sales and marketing, are quickly becoming digitized to increase efficiency and expedite execution. According to Gabryelczyk and Biernikowicz [1], organizations invest a significant effort to accomplish digitization, as it is one of the key aspects that lead to success.

Digitalization can also transform the way legal processes are carried out in modern times. It can create a more efficient and streamlined approach to legal proceedings, leading to increased accuracy, speed in decision making, improved transparency, accountability, and traceability of the process, while reducing overall costs, as described by Zorzanelli Costa et al. [2].

However, the percentage of digitized legal processes still lags behind that of business processes. Although procedural law is basically a definition of the processes that must be followed when complying with legislation, digitization of legal processes is much more challenging compared to an organization's processes. Legal processes are often subject to a complex web of regulations and compliance requirements that can be difficult to navigate when implementing digital solutions. Moreover, the intention to digitize processes was not usually included when proposing new laws or approving changes to legislation in the past, as written by Ciaghi and Villafiorita [3].

As a result, many projects dealing with digitization of the legal process are progressing slowly or becoming much more expensive. An example that can be mentioned is the Czech Republic. Although the government presented detailed plans describing the state administration's digitization strategy in 2018, only a small percentage of the proposed projects were completed, as referred to in the report by the Supreme Audit Office of the Czech Republic [4].

To summarize, despite the fact that digitization of state administration and legal processes can expedite execution and reduce costs, as business process digitization shows, the transformation to a way of working using systems as supporting tools for legal processes is slow.

In order to simplify the development process and eliminate the problems that come with the implementation of systems supporting legal processes, this thesis focuses on the possibilities that bring the involvement of the techniques and methodologies researched at the Czech Technical University.

The goal of the thesis is to create a Business Process Management (BPM) system – a system driven by process representation in order to coordinate the execution – that supports the tax administration process, one of the processes defined in the Czech code of law. To achieve this goal, the development process applies a specific approach that uses Design and Engineering Methodology for Organizations (DEMO) – modeling methodology for analysing and representing business processes – to analyse the current as-is state of the process, Business Process Model and Notation (BPMN) – graphical notation that depicts the steps in business processes – to design to-be state of the process after digitization, and Camunda Platform to implement the BPM system supporting the tax administration process based on the solution designed in the previous step.

The thesis consists of five chapters in total. The first chapter reviews the state-of-the-art of current practices, technologies, and methodologies. Law modeling and BPM systems are described, and an overview of BPMN and its constructs is given, along with the DEMO methodology and its rules. Lastly, the key abilities and the way of using the Camunda Platform are characterized.

The second chapter describes the as-is analysis of the tax administration process as defined in the Czech code of law. Organizational Essence Revealing (OER) analysis is used to reveal the process essence, identify transactions and actors, and mainly create the basis for the further presented DEMO diagrams.

Using the insights gained during the as-is analysis, the proposed to-be state of the tax administration process after digitization is designed. The process of designing the post-digitization state is explained, and the final result of the to-be analysis represented by the analytical BPMN model is shown in the third chapter.

The fourth chapter describes the realization of the BPM system using the Camunda Platform. As a part of the chapter, the hosting possibilities, the overall system architecture, and its main components are defined. Next, the creation of the executable BPMN model and the configuration of details necessary for correct deployment, as well as the implementation of the Spring Boot client, are specified. The end of the chapter contains an overview of the created system with an example of its usage.

Finally, the last chapter of the thesis summarizes the development process of the BPM system and expresses the key findings and the advantages and disadvantages of DEMO and BPMN involvement. The chapter ends with the description of the further development possibilities.

Chapter 1

Theoretical foundations

The first chapter of the thesis will focus on technologies, methodologies, and concepts that are well-known nowadays and can be used to produce systems supporting legal processes. Firstly, the motivation and the law modeling and its advantages and limitations will be briefly described. The next sections will overview BPM systems and summarize the current standards and constructs of BPMN. In the following section, the DEMO methodology will be presented and its key features will be pointed out. Next, Camunda will be introduced and its usage and main benefits will be explained. In the final section, a description of the development methodology used to create a system supporting legal processes will be provided.

1.1 Motivation

The legal system plays a vital role in the functioning of society. However, legal processes can often be slow, complicated, and expensive, leading to frustration and dissatisfaction among citizens. In recent years, there has been a growing trend towards the digitization of legal processes, with the aim of making them faster, more efficient, and more accessible to all.

According to a report by the Supreme Audit Office of the Czech Republic [4], the government spent 75 billion Czech crowns (approximately 3.19 billion euros) on digitization projects between 2012 and 2018. However, the current state of digitization in the country is not satisfactory. The creation of systems that condition digitization is progressing slowly, and the resulting solutions sporadically contain errors that make their use impossible [4].

An example that can be mentioned is the system dealing with vehicle registration. The total cost of the system increased to 47 million Czech crowns (2 million euros), but, despite this large investment, the Ministry of Transport of the Czech Republic considered returning to the old system a few days after deployment due to problems encountered with the new system, as described in [5]. Similarly, when a new census system was introduced at a cost of 30 million Czech crowns (1.3 million euros), it became unavailable soon after launch, as described in [6].

These examples are not just exceptions to otherwise problem-free systems. In summary, a considerable portion of the Czech Republic's budget is spent on digitization, yet the resulting systems are not always of high quality. Although the government presented a detailed plan describing the digitization strategy in 2018 [7], as outlined in the report by the Supreme Audit Office [4], progress is slow. A more effective solution for creating systems is necessary to keep up with future needs and expedite the digitization process.

1.2 Law modeling

In recent years, modeling the semantics of law has gained a notable amount of attention. According to Ciaghi et al.: [8]: "*Providing a graphical representation of a law can be of great advantage to those who want to understand or analyse it (e.g., citizens or jurists) as well as those who need to implement it.*" Moreover, law modeling plays a key role in the automation of state administration.

The way automation can be achieved is similar to the automation of processes that are modeled to analyse and improve the behavior of an organization in the private sector. Since legal systems encroach on many aspects of our lives, their functioning demands a much higher level of transparency than what is required of other organizational environments such as private enterprises, as described by Zorzanelli Costa et al. [2].

However, legal documents often rely on opaque aspects of legal jargon used to define legislation or on a number of procedural and operational aspects embodied in practice that are not explicitly captured in legal documents. Furthermore, the legal system usually creates a complex structure that can sometimes be impossible to precisely capture using ambiguous natural language. These aspects create several barriers to transparency that is necessary for citizen access to justice as well as the design and operation of the digital legal information system, as written by Zorzanelli Costa et al.[2].

Modeling law brings transparency to legal documents and aids in the digitization of the legal system. However, according to Ciaghi and Villafiorita [3], it is necessary to link the modeled procedures to the regulations that define and direct them. In certain situations, it may not be possible to create a model based on the current legal definition, and the modeling process may require parallel actions in both the process design and the introduction of legal changes.

1.3 BPM systems

To create a solution that helps with automation of process, a Business Process Management (BPM) system can be used. Van der Aalst [9] writes that "*Business Process Management (BPM) includes methods, techniques, and tools to support the design, enactment, management, and analysis of operational business processes.*" A business process is a set of activities that are performed in an organized way, usually using a technical solution, with the purpose of achieving a specified business goal.

Traditionally, business processes were managed manually, guided by the knowledge of the organization's personnel who followed the regulations and rules of the organization's procedures. In recent years, organizations have discovered that they can achieve additional benefits by using software solutions to coordinate the activities of business processes, as described by Weske [10]. These software solutions are called Business Process Management systems. Weske [10] defines BPM system as "*a generic software system that is driven by explicit process representations to coordinate the enactment of business processes.*"

According to Gabryelczyk and Biernikowicz [1], organizations are adopting BPM systems because they bring numerous benefits. As examples from the past show, the involvement of BPM system usually results in the improvement of process efficiency, reduction of process costs, as well as the elimination of process errors, and improvement of collaboration within the organization.

1.4 BPMN

Business Process Model and Notation (BPMN) is a standard for business process modeling that enables businesses to comprehend their internal processes and structure through a graphical notation. The primary benefit of BPMN is its flexibility. The models can be simple enough for stakeholders responsible for designing, managing, and implementing business processes to understand them. On the other hand, models can contain enough information to be translated into software components. The difference is in the selected level of detail. BPMN acts as a bridge between business analysts who draft processes, developers who implement information systems supporting the processes, and business people who manage and monitor the processes, as described in [11].

According to Malekan et al. [12], a major drawback of BPMN is its ambiguity in process execution. The realization of modeled constructs can be open to interpretation and the lack of concrete semantics can result in different implementations of the same modeled process by different developers.

1.4.1 Development and usage of BPMN

BPMN was developed and published in 2004 by the Business Process Management Initiative (BPMI) with the aim of creating a single standard notation that combined the best ideas from various divergent notations, such as UML Activity Diagrams and Event-Process Chains (EPCs). This is why there are noticeable similarities between BPMN and, for example, UML Activity Diagram. According to Alweyer [13], BPMN has since become the most widely accepted and used standard for business process modeling. Today, BPMN is maintained and developed by the Object Management Group (OMG), which is also responsible for other standards, such as the aforementioned UML.

Alweyer [13] also notes that BPMN is not only well-established among organizations as their business process modeling standard, but it can also be used by state administrations as a tool to help with digitization. Currently, BPMN plays a key role in Switzerland's e-government, as described by Walser and Schaffroth [14]. Standards of Switzerland's e-government specify BPMN as recommended notation as it helps with the integration of local and federal tasks into the process chain, where collaboration can be better designed. Other examples mentioned by Alweyer [13] include public administrations in Queensland, Australia, and British Columbia, Canada, where BPMN creates guidelines for their processes.

1.4.2 BPMN constructs

The current version of BPMN, BPMN 2.0, was released by OMG back in 2011. In order to create a simple and understandable mechanism for capturing business processes, OMG [11] defined five basic categories with several graphical elements that represent the building blocks of the BPMN models. These categories are Flow objects, Data, Connecting objects, Swimlanes, and Artifacts.

1.4.2.1 Flow objects

Flow objects are the most important graphical aspects that define the behavior of the modeled business process. The OMG specification of BPMN 2.0 [11] describes three Flow objects:

- Events
- Activities
- Gateways

According to the official specification [11], "*an event is something that "happens" during the course of a process. . . . These events affect the flow of the model and usually have a cause (trigger) or an impact (result).*"

An activity describes any work that happens in the organization's process. BPMN specifies two types of activities that represent parts of the process. The activity can be either a task or a sub-process.

A gateway is used to determine the divergence and convergence of the sequence flow in the process. By the gateway branching, forking, merging, and joining of paths is controlled the overall process flow.

1.4.2.2 Data

Data category represents the information that is part of the process. Elements of this category are Data objects, Data inputs, Data outputs, and Data stores [11]. Data object provides information about the requirements of activities. Without the singular object or a collection of objects described by Data objects, the activity cannot be performed. The same principle applies for Data inputs and Data outputs.

Data stores provide a mechanism for an activity to retrieve or update stored information that is persisted outside the scope of the process. They can be used to model databases, file systems, document archives, and other data storage systems.

1.4.2.3 Connecting objects

A Connecting object defines the way of connecting the Flow objects to each other or to other elements. They play an important role in describing the flow of a business process and help to visually represent the dependencies between different elements. There are four types of Connecting objects defined in the specification of BPMN 2.0 [11]:

- Sequence flow
- Message flow
- Association
- Data association

Sequence flow is a type of Connecting object used in BPMN to define the order of the activities in a process execution. It represents the flow between two Flow objects. In other words, it defines the direction in which the process flows and specifies the order in which the events, gateways, and other Flow objects in the process are executed.

Message flow represents the flow of messages between two participants in a process. The message flow element points from the participant that is prepared to send a message to the participant prepared to receive one.

An association is used to show the linkage of Artifacts with the BPMN graphical elements. The same principle applies for data associations, with the only difference being that the connection is between Flow object and Data element.

1.4.2.4 Swimlanes

Swimlanes are used to group the modeled elements. In BPMN, Swimlane is either a lane or a pool [11]. The pool represents a participant or it can also be used as a graphical container that partitions a set of activities. The lane is a sub-partition in a process (often within the pool) that is used to categorize and organize the activities.

1.4.2.5 Artifacts

Artifacts are used to provide additional information about the process itself. Currently, the specification [11] defines two types of standardized Artifacts (Group and Text annotation), but modelers are free to add other necessary Artifacts.

1.4.3 BPMN levels

As mentioned before, the main advantage of BPMN is its flexibility, which enables the creation of less or more detailed models. Since the area of BPMN application is diverse, modelers who come into contact with the notation do not have to use or know all the elements. Even though BPMN has constructs that suit well for exception handling, modelers may find them unnecessary or even useless because they do not see the need to add them to their process model. The same applies to many other constructs. In fact, only a small number of elements are commonly used, as described by Silver [15].

Because of that, Silver [15] defines three levels of BPMN use:

- Level 1: Descriptive BPMN
- Level 2: Analytical BPMN
- Level 3: Executable BPMN

BPMN is mostly used at level 1, where descriptive modeling aims to document the process flow in a simple way using a limited number of constructs that are easily understood by business users. The result is similar to traditional flowcharting. Despite this, level 1 is sufficient for describing most processes and their behavior at the level that businesses need.

Level 2 includes additional constructs that enable more accurate process modeling, taking into account exceptions and events. According to Aagesen and Krogstie [16], "*The additional features are particularly relevant to include when doing computer-assisted analysis, supporting quality assurance, and when the models are meant to be used as context for change through a traditional development project.*"

Both descriptive and analytical BPMN concern non-executable processes, and their benefit relies solely on the information visible from the modeled process. On the contrary, BPMN level 3 is based on XML details that are typically not visible directly from the model. The main purpose of executable BPMN is its ability to be transformed into an XML-based specification that can drive process engines, as explained by Aagesen and Krogstie [16].

1.5 DEMO methodology

Design and Engineering Methodology for Organizations (DEMO) is a methodology that is used to construct enterprise models based on the Enterprise Ontology, as defined by Dietz and Mulder [17]. The main goal of the methodology is to give a modeler an overview of and insight into an organization. To capture the essence of an organization and its processes, DEMO methodology presents the essential model [18]. The essential model of a system or an organization is described as an ontological model, in other words, a model of its construction that is completely abstracted from implementation.

The current version of DEMO, DEMO-4, is based on theoretical foundations that consist of four ontological and three philosophical theories, all referred by Greek letters [19]. According to Dietz and Mulder [17] "*philosophical theories concern the most fundamental ways in which people perceive and conceive the surrounding world, make sense of it, study it, etc.*" On the other hand "*ontological theories are about the nature of things. They serve to explain their construction and operation, and predict the consequences of changing them, while completely abstracting from implementation*" [17].

Because BPMN due to the heterogeneity of its constructs lacks formal semantics, which can lead to inconsistency and ambiguity of the modeled business process, DEMO and the Enterprise Ontology can be applied to add a formal foundation to BPMN model creation, as recommended by Van Nuffel et al. [20].

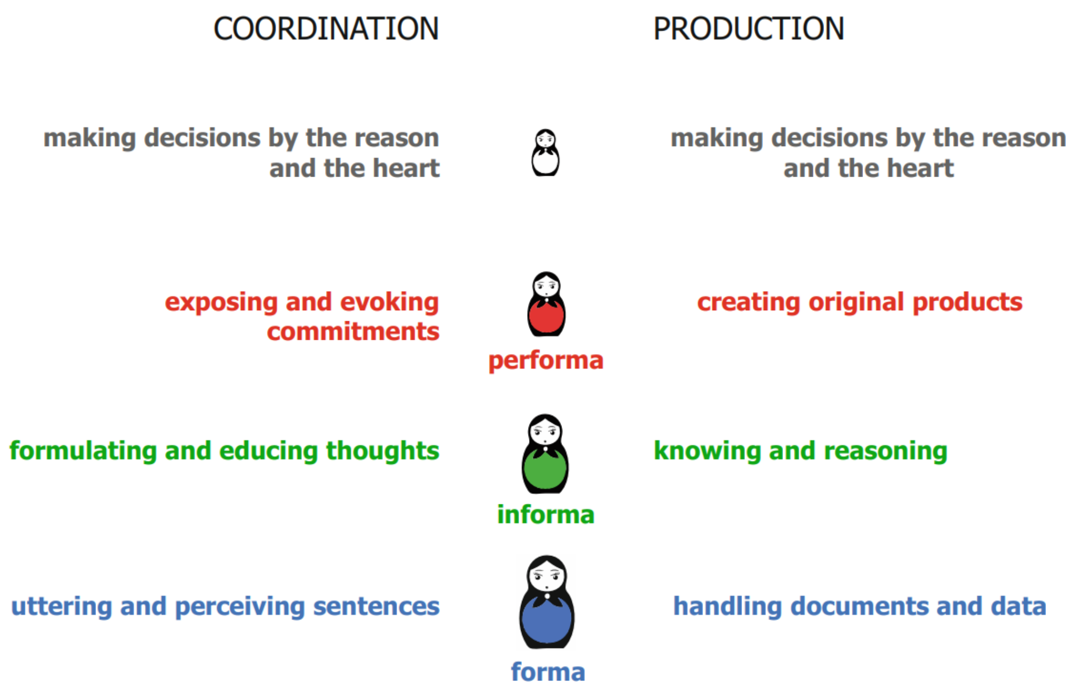

COORDINATION
PRODUCTION
making decisions by the reason and the heart
making decisions by the reason and the heart
exposing and evoking commitments
performa
creating original products
formulating and educing thoughts
informa
knowing and reasoning
uttering and perceiving sentences
forma
handling documents and data

Figure 1.1 The abilities in coordination and production acts [17].

1.5.1 OER analysis

According to Dietz and Mulder [17], in a running organization or process, no one needs to devise or create an essential model from scratch using any other technique. The way an organization works, its operational essence, already exists, and it only needs to be revealed.

The best way of revealing and understanding the essence of an organization is to address the people working there, which basically means interacting with the actors themselves. Another option is to rely on written documentation. The main reason why it is not considered the best option is the difference between description and reality, no matter how thoroughly the documentation is written. Although written documentation analysis is not the best way to recognize the hidden essence of an organization, in practice it is the most widespread.

The method in DEMO that is used to reveal the essential model of a running organization or process is called OER analysis. OER stands for Organizational Essence Revealing. The starting point of the OER analysis is written documentation, as it is used in most cases of revealing organizational essence. The whole method consists of four steps [17].

The first step of the OER analysis is used to determine the transaction acts and corresponding information in an organization. To achieve the goal, distinguishing between the performa, informa, and forma shapes occurs. That is the reason why step one of the OER analysis is also called the performa-informa-forma (PIF) analysis. To be able to successfully complete the PIF analysis, one needs to understand that all acts that are part of a process or an organization can be divided into two sorts: production acts and coordination acts. In those acts, three abilities are distinguished: forma ability, informa ability, and performa ability. A closer view with short descriptions of these abilities can be seen in Figure 1.1. To emphasize the fundamental humanness of actors, the first, most inner ability is added. At the completion of the first step of the OER analysis, parts of the existing documentation should be highlighted if they express performa, informa, or forma matter.

The second step of the OER analysis consists of identifying transaction kinds and actor roles. Dietz and Mulder [17] define an actor as "*a subject (human being) in an actor role. The actor role determines the authority that the actor may exercise and the responsibility to do so.*" Coordination acts defining the interaction between actors, which play either an initiator or executor role, occur in a certain pattern that can be seen in Figure 1.2. These acts can be divided into four basic types: request, promise, declare, and accept and two additional types: decline and reject. The complete transaction pattern also contains the revocation of basic types. Every transaction or its instance represents the transaction kind that concerns one specific product kind. To summarize the above, the result of the second step of OER analysis is a detailed description of the transaction kinds representing a transaction that is, in fact, a sequence of coordination acts in a specific pattern. Each transaction has its initiator and executor, who is responsible for creating the product. The combination of a transaction kind and its executor is termed transactor.

After the first two steps, the transaction types and actor roles are identified. The third step of the OER analysis consists of building the essential model of an organization. Typically, the scope of interest is narrowed or broadened in this step so that the transactor roles correspond to the identified transaction types. This means that the first two steps may need to be reworked to find missing or to leave out redundant transactor roles. The main goal of this step is to understand the interaction structure of an organization that is defined by a number of tree structures. Unlike in business process management, where the flow type of thinking is mostly used, in DEMO the process is modeled from a structural perspective. The tree diagram is ideal for representing such a structure because of its ability to capture the composition of the identified transaction types. Once the process structure is defined, it is possible to create Cooperation Model, Action Model, Process Model, or Fact Model described more thoroughly in the following section.

Some works, such as that written by Perinforma (pseudonym of Dietz) [21], list the third step of the OER analysis as the final one. If the previous steps are followed, the result also complies with most of the techniques used to achieve essence and simplicity in DEMO, including separation of concerns, use of abstraction, devising proper concepts, and verification by instantiation. Only validation from ontology does not immediately emerge from the following of the previous three steps. Therefore, the fourth step is added, which consists of validating the essential model. The procedure recommended by Dietz and Mulder [17] for the fourth step is "*to study the identified transactor roles one by one and to check the claims that the model implicitly makes*". After that, a verified and validated essential model within the DEMO methodology can be produced.

1.5.2 Models

As mentioned in the previous section, DEMO, more specifically DEMO-4, defines four aspect models. These aspect models can be used individually or in combination to analyse and design different aspects of an organization. Together, they provide a comprehensive and integrated view of an organization, which can help to identify and resolve issues related to organizational design, communication, and coordination. The relationships between the aspect models and their main concerns are illustrated in Figure 1.3.

Dietz and Mulder [17] define the aspect models and describe their main benefits in revealing the process essence as follows:

- **Cooperation Model (CM)**

  The Cooperation Model captures the cooperative interactions between people and organizations. The model is typically represented using two primary artifacts: the Coordination Structure Diagram (CSD) and the Transaction Product Table (TPT). The CSD depicts the actors involved in the cooperative interactions, the transactions between them, and the roles

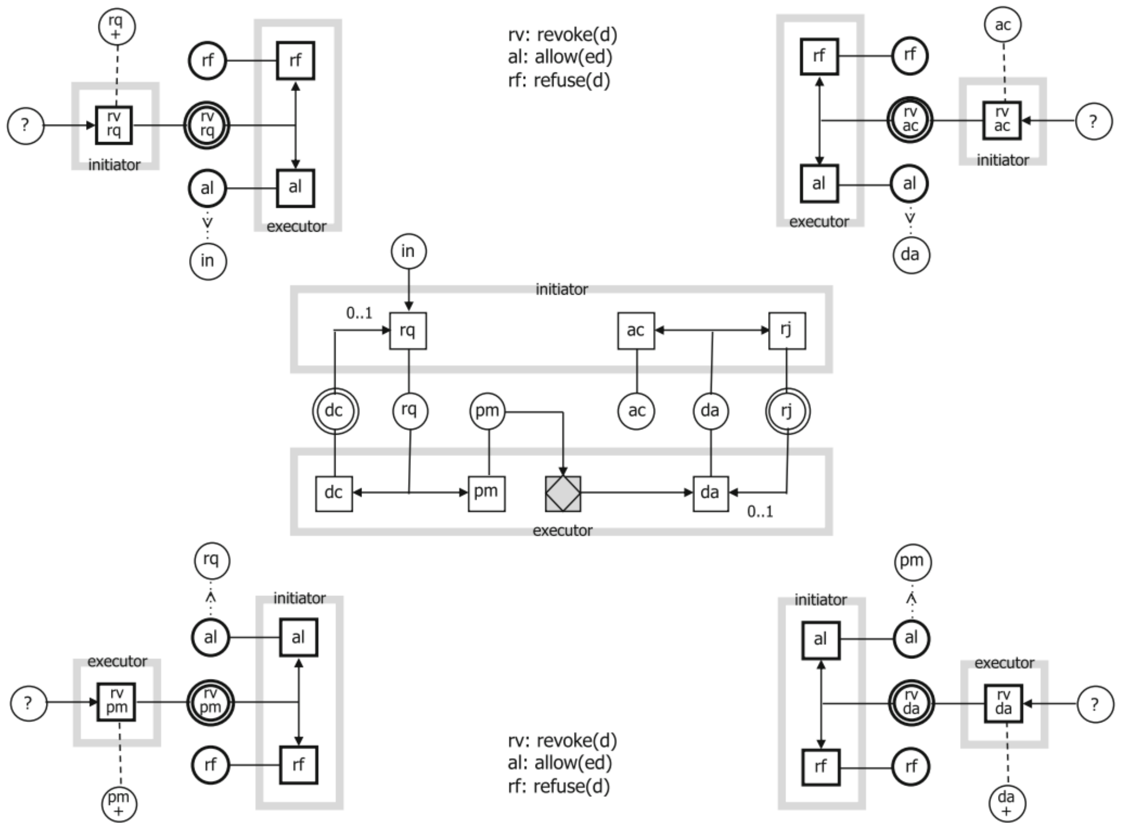


Figure 1.2 The complete transaction pattern [17].

and responsibilities of each actor. The TPT, on the other hand, shows the products and a detailed description of the coordination acts between the actors and the associated commitments, constraints, and dependencies. Together, these artifacts provide a comprehensive view of the cooperative interactions.

- **Action Model (AM)**

  The Action Model describes the manifestation of the construction of an organization over time. The Action Model is conveyed through an Action Rule Specification (ARS) and a Work Instruction Specification (WIS).

- **Process Model (PM)**

  The Process Model is a model of the processes that occur as a result of actions taken by actors. The Process Model is used to create a description of the state space and transition space of the coordination world. The Process Structure Diagram (PSD) is used to express the Process Model of an organization.

- **Fact Model (FM)**

  This model is used to define and visualize the products of an organization. It specifies the state space and transition space of the production process. The Object Fact Diagram (OFD) is a representation of the Fact Model.

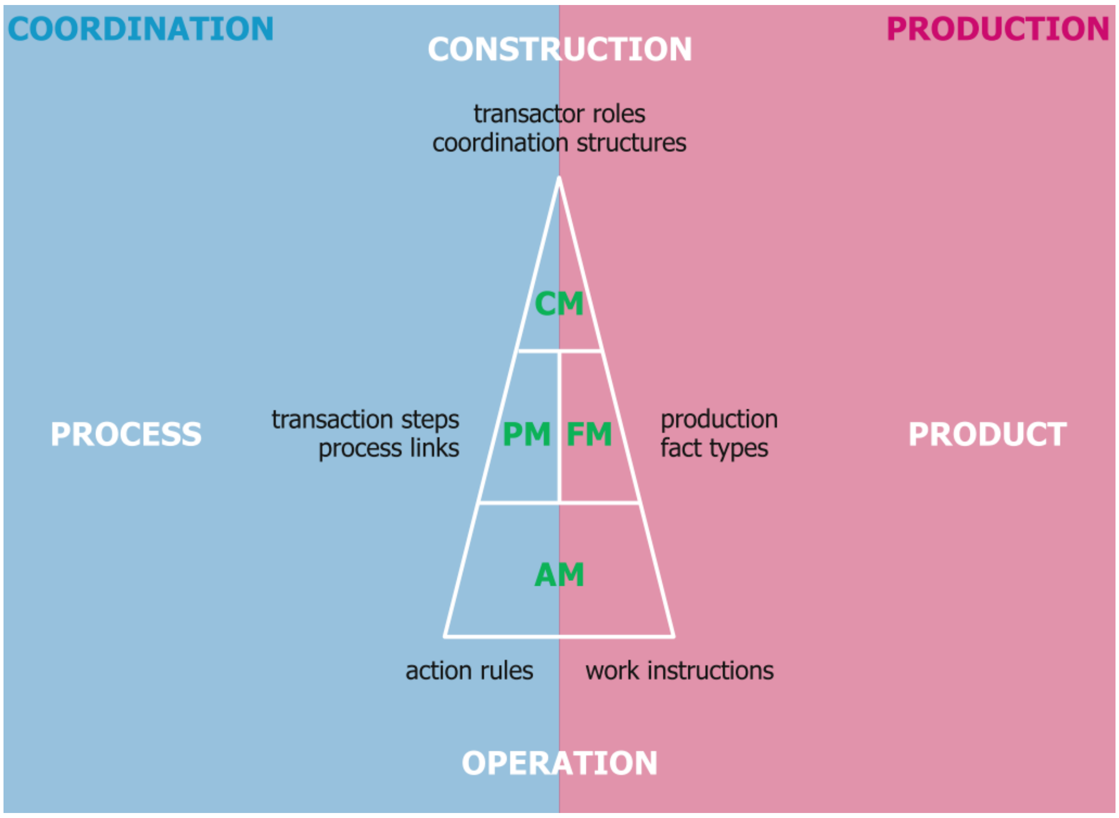


Figure 1.3 The relationships between aspect models and their concerns [17].

1.6 Camunda

Camunda is a platform that uses a workflow engine and a decision engine to orchestrate and automate complex business processes that span people, systems, and devices. Organizations apply the Camunda Platform solution to model and automate workflow and decision processes using BPMN-powered flowcharts as defined in the documentation [22].

To put it into perspective, one of the main components of Camunda is the engine that can be used to run executable BPMN models, which helps developers with creation of business process management tools that orchestrate the tasks necessary for process accomplishment.

1.6.1 Microservices

To run an automated end-to-end business process, it typically requires multiple microservices to achieve the desired outcome. According to Fowler and Lewis [23], the microservice architecture is basically "*an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms.*" Each of these services is built around business capabilities, and the process of their deployment is independent and usually fully automated by the deployment machinery.

However, with the use of microservice architecture often comes the struggle of developers to effectively communicate across the services, monitor their performance, or resolve the errors that may occur during runtime. The newest version of the Camunda Platform – Camunda Platform 8 – enables organizations to overcome these issues [22].

Since microservice architecture has rapidly gained popularity in the past few years, as described, for example, by Dragoni et al. [24], this thesis will mainly focus on the application of the newest Camunda Platform 8, which is based primarily on the orchestration of tasks that require human involvement alongside microservices as written in [22].

1.6.2 Camunda Platform 8

Camunda Platform 8 was released in 2022 as a successor to the already used Platform 7. Its functionality is specifically designed to "*orchestrate microservices into trackable and manageable business processes without compromising crucial microservices principles such as loose coupling and service independence,*" as defined in [25]. Jakob Freund [26] describes the release of Camunda Platform 8 as a step toward "*a vision of the universal process orchestrator, allowing one to orchestrate people, systems, and devices along BPMN process models that bring business and IT together, and doing this reliably and at scale*".

This implies that Camunda Platform 8 is designed to operate on a large scale. To achieve this, Camunda Platform 8 provides following abilities that ensure core quality, as defined in the documentation [22]:

- **Horizontal scalability**

  Zeebe (the workflow engine that is used in Camunda Platform 8, more thoroughly described in the chapter *Implementation of proof-of-concept BPM system using Camunda Platform*) allows to run the process without dependence on an external database. It has the ability to write data directly to the file system on the server, where it is deployed. The application of Zeebe as a workflow engine has a positive effect on the simplicity of distributing processing across the cluster to ensure high throughput.

- **High availability and fault tolerance**

  Camunda Platform 8 has a built-in, pre-configured replication mechanism, which makes it possible to recover from a failure with no data loss and minimal downtime. The replication process is fully automated, which means that no manual action is required.

- **Audit trail**

  All process-relevant information is written into an append-only log that can serve as a detailed history of the process if necessary.

- **Reactive publish-subscribe interaction model**

  The model enables microservices to connect to the Camunda Platform 8 while maintaining autonomy and a high degree of control. These properties allow for scalability, resilience, and reactiveness.

- **Visual processes modeled in ISO-standard BPMN 2.0**

  As defined before, the use of the BPMN standard ensures the possibility for both technical and non-technical stakeholders to collaborate on a process design due to the ease of understanding the notation.

- **Language-agnostic client model**

  This ability makes it possible to build a client in a wide range of programming languages to suit the developers of an organization.

1.7 Methodology

As the state-of-the-art BPM systems, BPMN, DEMO, and Camunda were reviewed, the approach that can be used to develop a BPM system can be presented. The methodology that aims to create a system supporting legal processes consists of three phases:

- As-is analysis
- To-be analysis
- Implementation

The first phase uses DEMO methodology to reveal the essence of the procedural part of the law. The OER analysis and DEMO models are used to analyse and capture the as-is state of the process based on its definition in the code of law. DEMO's focus on essential elements and structures makes it suitable for legal processes, which typically involve a high level of complexity and detail. According to Van Nuffel et al. [20], the involvement of DEMO methodology can give a formal foundation for further BPMN modeling.

The to-be analysis phase involves using an analytical BPMN model to design the future state of the process. As BPMN is a widely used process modeling notation, it provides an understandable way of representing the to-be state graphically. The analytical BPMN model is a detailed model that includes useful information when capturing the state of the process after digitization. According to van der Aalst [27], analytical BPMN models can be used to simulate and optimize processes before implementation. This implies that the approach can ensure the efficiency and effectiveness of the resulting BPM system.

The final phase corresponds to the implementation of the BPM system. It involves using the Camunda Platform to implement the system based on the executable BPMN model, which is an extension of the analytical model from the previous phase with implementation aspects. As the Camunda Platform contains an engine that can run the executable model, the overall need to write the code during the implementation phase can be reduced, as described by Silver [28].

In conclusion, the methodology that aims to create a BPM system supporting legal processes involves as-is analysis using DEMO methodology and its OER analysis, to-be analysis using an analytical BPMN model, and implementation based on an executable BPMN model and Camunda Platform. The methodology, based on the mentioned assumptions, ensures that the resulting system is designed to be an effective and useful supporting tool during the process execution.

Chapter 2

Tax administration as-is analysis

To convert the assumptions described in the chapter *Theoretical foundations* into practical use and prove the concept, a case study based on existing law is included as part of the thesis. The case study is mainly focused on verifying the selected methodology and its potential for use in the digitization of the law.

The case study contains a description of the DEMO as-is analysis, followed by a to-be analysis based on an analytical BPMN model, and the creation of a BPM system that leads to the digitization of the law as defined by the methodology. The Czech tax administration law is used as an example to illustrate this development process.

This chapter will focus on analysing the current state of the tax administration process, as described in the Czech code of law. DEMO and its OER analysis will be the main tools used to reveal the essence of the tax administration process, just as it does with the essence of an organization's processes. The goal of the as-is analysis is to understand the process workflow and add the formal foundations to the BPMN models that will be shown and described in detail in the next chapter.

First, a brief overview of tax administration will be provided. Following that, the four steps of the OER analysis will be applied to reveal the main parts of the analysed law. In the following sections, the resulting diagrams that are important for further development will be shown, and the main modeling thoughts will be closely described. Finally, at the end of the chapter, a short summary with the key findings of the tax administration as-is analysis will be presented.

2.1 Tax administration overview

This section provides a summary description of the tax administration as defined by the Czech code of law, with a specific focus on its procedural part that outlines the process of creating an official decision, which is the objective of the entire tax administration.

The tax administration is described in detail in Act No. 280/2009 Coll., Tax Administration [29]. The procedural part of the law that is particularly relevant to the case study presented in this thesis is defined in paragraphs § 91 – § 107. In addition to defining the tax administration process, the act establishes the rights and obligations of the tax subject as well as obligations of the tax administrator.

The tax subject and the tax administrator are the primary parties involved in the tax administration process. The tax subject is the person whose tax is determined during the process and is obligated to cooperate with the tax administrator. The tax administrator is responsible for overseeing the process to achieve its goal.

As mentioned above, the ultimate goal of the tax administration process is to create an official decision that specifies the final amount of tax, along with the justification and reasons for the tax assessment. This official decision serves as a binding document that is valid from the day of its adoption.

2.1.1 Tax administration process

The process of creating an official decision is initiated on the day that the tax administrator receives a submission from the tax subject. To ensure that the tax is determined correctly, the tax subject is asked to provide facts that will lead to an accurate determination of the tax. The administrator then decides whether the submitted facts are sufficient for the tax determination. If the facts are sufficient and no additional necessary information is required, the final decision can be created and sent to the tax subject.

However, if the submitted facts are not sufficient to determine the tax correctly, the tax administrator is authorized to initiate the creation of an expert report, witness testimonies or to obtain other additional information in order to create the decision.

Once the official decision is created, the tax subject is obligated to pay the costs of the proceedings in case any expenses are incurred. These costs are related to the course of the process. For example, if it is necessary for a third person to participate in the proceedings to properly determine the tax, the participant may be awarded compensation. The entitlement of the proceedings participant to compensation, as well as its final amount, is assessed by the tax administrator.

The process ends when the costs are paid and the official decision, along with all of its determining information and reasoning, is published. Part of the publication process involves sending the decision to the tax subject as well as saving it to the appropriate system. Based on the determined amount of tax, the subject also receives the payment obligation.

2.2 OER analysis

The following sections will focus on the individual steps of the OER analysis defined by the DEMO methodology. The starting point of the OER analysis is the Czech code of law, specifically the description of the tax administration process, as it can be considered a kind of documentation that defines the process to be revealed.

2.2.1 Distinguishing performa-informa-forma

The first step of the OER analysis, as described previously, consists of performing the performa-informa-forma (PIF) analysis. To distinguish the found performa, informa, or forma abilities in the text, each one is highlighted with a different color. Performa ability is highlighted using the color red, informa using the color green, and the color blue is used for forma ability, following the established practice among DEMO modelers.

The text also contains a significant amount of so-called *blue traps*. The term *blue trap* refers to a part of the text that, at first glance, expresses the forma ability, but also has a hidden meaning in the form of the performa ability, as described by Jan Dietz [17]. Because of the hidden performa meaning, the part of the text is not highlighted using the blue color, but instead, the color red is used.

The resulting text of the Czech code of law with highlighted parts that match the abilities can be accessed in the repository corresponding to the thesis.

2.2.2 Identifying transaction kinds and actor roles

The next step of the OER analysis consists of identifying transaction kinds and actor roles. Since the text contains highlighted performa abilities, the act types can be assigned to them. Each of the performa abilities in the text should be marked as one of the coordination act types, such as request – rq, promise – pm, declare – da, accept – ac, decline – dc, reject – rj, (or revokes of basic types – rvrq, rvpm, rvda, rvac), or as a production act representing the creation of the transaction product.

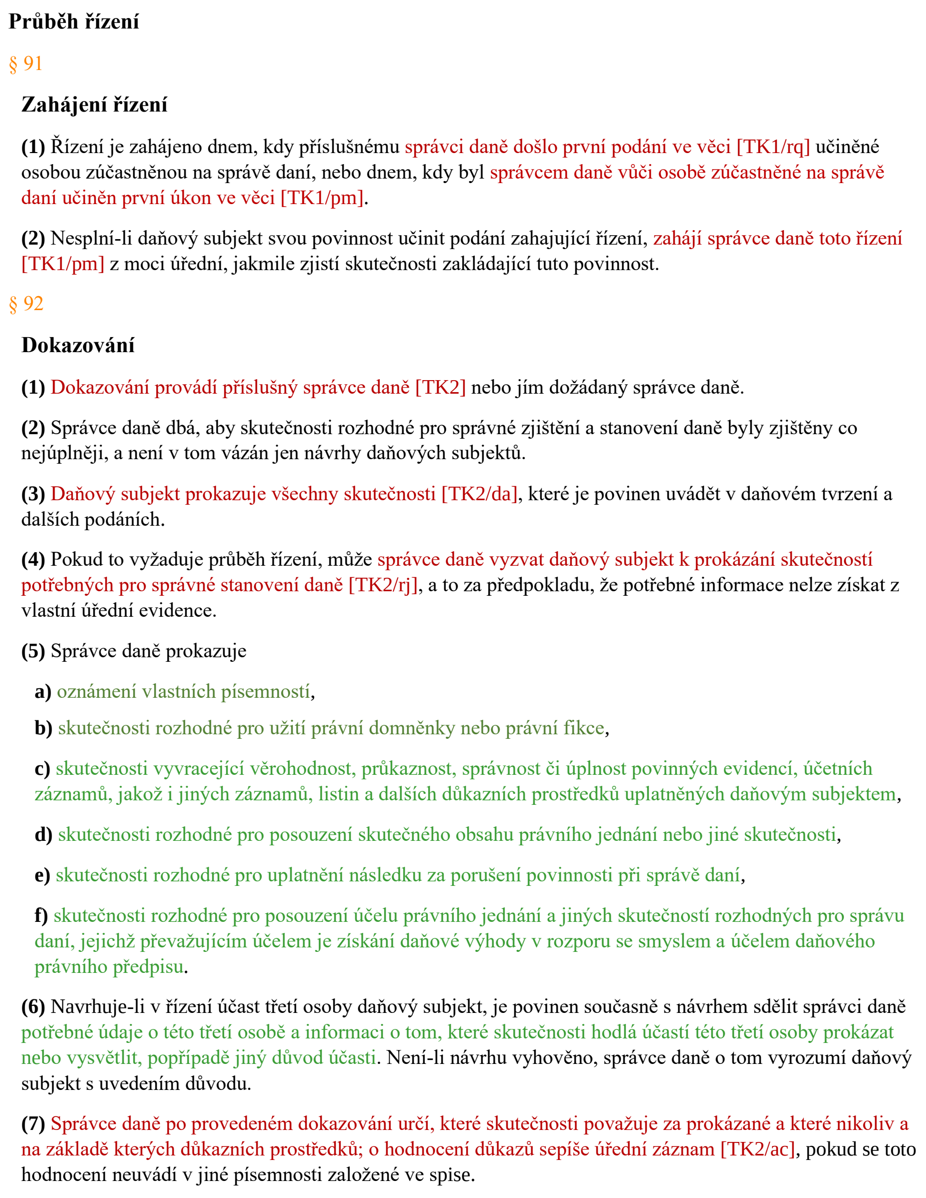
Based on this, transactions are recognized in the law text. An example of highlighted text with assigned acts of the recognized transactions can be seen in Figure 2.1. Specifically, the example contains paragraphs § 91 and § 92, which mainly describe the initiation of the tax administration proceedings and the process of validating the submitted facts. In the example, two transactions TK1 and TK2 can be seen that correspond to the official decision creation process and the validation of the submitted facts process, respectively. The same way that the text of the tax administration law is analysed in the example, the rest of the text is also analysed. The whole text with identified transactions can be seen in the repository corresponding to the thesis.

Overall, the tax administration process consists of 11 transactions. Found transactions are expressed using extended Transaction Result Tables (e-TRT), which contain, for each transaction, the corresponding initiator and executor, transaction product, as well as all found coordination acts. An example of the e-TRT that describes the official decision creation (TK1) can be seen in Table 2.1. The rest of the e-TRT is shown in the mentioned repository.

Transaction	Official decision creation (TK1)
Product	Official decision is created
Initiator	Official decision creation proposer (AR1)
Executor	Official decision creator (AR2)
Request	Submission application for decision creation
Promise	Initiation of official decision creation
Decline	Not specified
Declare	Sending created official decision
Reject	Not specified
Accept	Not specified
Revoke request	Withdraw submission
Revoke promise	Declaration of non-necessity of decision
Revoke declare	Proposing correcting decision
Revoke accept	Not specified

Table 2.1 Example of extended Transaction Result Table (e-TRT).

As described above, each transaction has its initiator and executor, both of which are termed actors. Since DEMO has specifically defined rules describing the naming of the actors, the actor names usually differ from the already used common naming. To express the connections, found actor roles and their correspondence to real-world subjects are visible in the Subject-Actor Table shown in Tables 2.2 and 2.3. Each subject that can be part of the tax administration process has defined roles that he can perform.



Průběh řízení

§ 91

Zahájení řízení

(1) Řízení je zahájeno dnem, kdy příslušnému správci daně došlo první podání ve věci [TK1/rq] učiněné osobou zúčastněnou na správě daní, nebo dnem, kdy byl správcem daně vůči osobě zúčastněné na správě daní učiněn první úkon ve věci [TK1/pm].

(2) Nesplní-li daňový subjekt svou povinnost učinit podání zahajující řízení, zahájí správce daně toto řízení [TK1/pm] z moci úřední, jakmile zjistí skutečnosti zakládající tuto povinnost.

§ 92

Dokazování

(1) Dokazování provádí příslušný správce daně [TK2] nebo jím dožádaný správce daně.

(2) Správce daně dbá, aby skutečnosti rozhodné pro správné zjištění a stanovení daně byly zjištěny co nejúplněji, a není v tom vázán jen návrhy daňových subjektů.

(3) Daňový subjekt prokazuje všechny skutečnosti [TK2/da], které je povinen uvádět v daňovém tvrzení a dalších podáních.

(4) Pokud to vyžaduje průběh řízení, může správce daně vyzvat daňový subjekt k prokázání skutečností potřebných pro správné stanovení daně [TK2/rj], a to za předpokladu, že potřebné informace nelze získat z vlastní úřední evidence.

(5) Správce daně prokazuje

a) oznámení vlastních písemností,

b) skutečnosti rozhodné pro užití právní domněnky nebo právní fikce,

c) skutečnosti vyvracející věrohodnost, průkaznost, správnost či úplnost povinných evidencí, účetních záznamů, jakož i jiných záznamů, listin a dalších důkazních prostředků uplatněných daňovým subjektem,

d) skutečnosti rozhodné pro posouzení skutečného obsahu právního jednání nebo jiné skutečnosti,

e) skutečnosti rozhodné pro uplatnění následku za porušení povinnosti při správě daní,

f) skutečnosti rozhodné pro posouzení účelu právního jednání a jiných skutečností rozhodných pro správu daní, jejichž převažujícím účelem je získání daňové výhody v rozporu se smyslem a účelem daňového právního předpisu.

(6) Navrhuje-li v řízení účast třetí osoby daňový subjekt, je povinen současně s návrhem sdělit správci daně potřebné údaje o této třetí osobě a informaci o tom, které skutečnosti hodlá účastí této třetí osoby prokázat nebo vysvětlit, popřípadě jiný důvod účasti. Není-li návrhu vyhověno, správce daně o tom vyrozumí daňový subjekt s uvedením důvodu.

(7) Správce daně po provedeném dokazování určí, které skutečnosti považuje za prokázané a které nikoliv a na základě kterých důkazních prostředků; o hodnocení důkazů sepíše úřední záznam [TK2/ac], pokud se toto hodnocení neuvádí v jiné písemnosti založené ve spise.

Figure 2.1 Example of analysed text after completion of the OER analysis.

	Tax administrator	Tax subject	Expert	Witness	Third person participating proceedings	Law enforcement
Official decision creation proposer (AR1)		X				
Official decision creator (AR2)	X					
Facts submitter (AR3)		X				
Expert report creator (AR4)			X			
Testimony attestant (AR5)				X		
Special records creator (AR6)		X				
Tax negotiator (AR7)		X				

Table 2.2 Subject-Actor Table of tax administration process, part 1.

	Tax administrator	Tax subject	Expert	Witness	Third person participating proceedings	Law enforcement
Proceedings participant (AR8)					X	
Summons executor (AR9)						X
Participant attachment executor (AR10)						X
Entitlement to compensation decisive person (AR11)	X					
Cost of proceedings payer (AR12)		X				

Table 2.3 Subject-Actor Table of tax administration process, part 2.

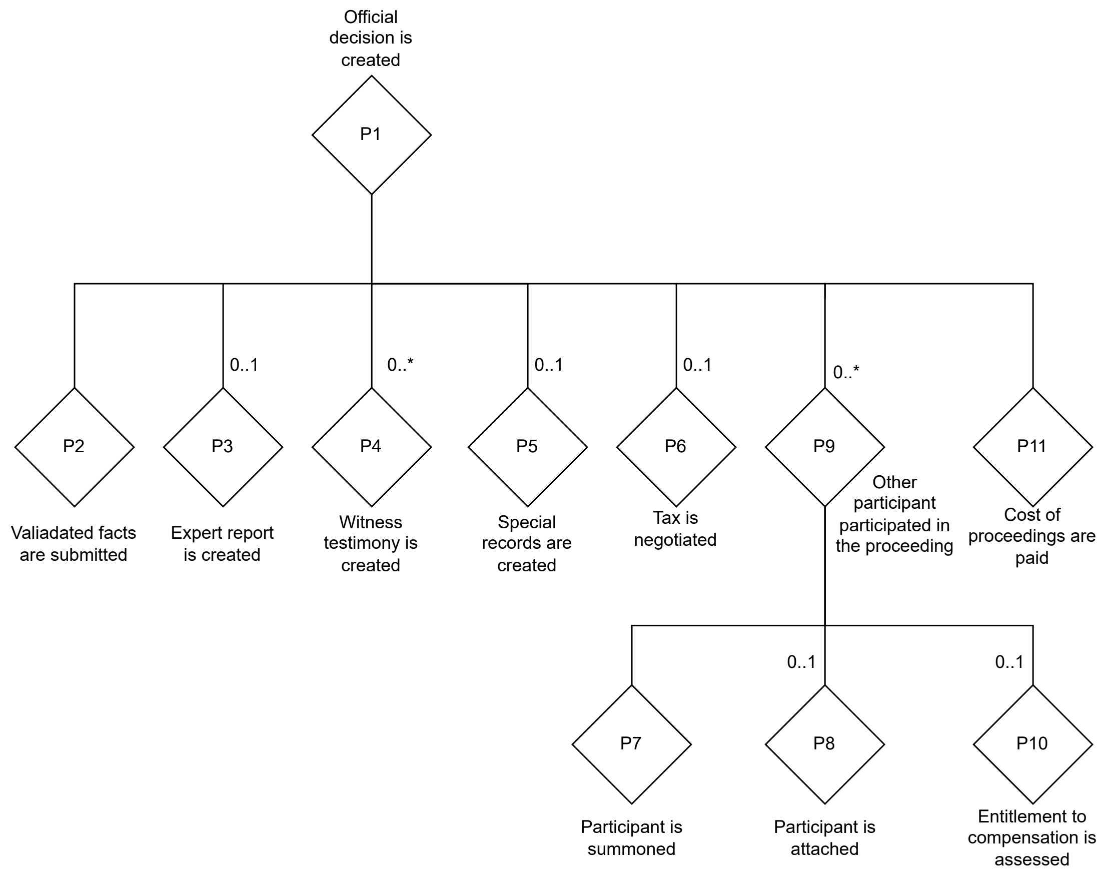

Official decision is created
P1
0..1
0..*
0..1
0..1
0..*
P2
P3
P4
P5
P6
P9
P11
Valiadated facts are submitted
Expert report is created
Witness testimony is created
Special records are created
Tax is negotiated
Other participant participated in the proceeding
Cost of proceedings are paid
0..1
0..1
P7
P8
P10
Participant is summoned
Participant is attached
Entitlement to compensation is assessed

Figure 2.2 Interaction Structure Diagram of tax administration process.

2.2.3 Composing the essential model

The third step is based on the composition of the essential model. As mentioned above, the essential model is used to understand the interaction structure of an organization or, in this case, the law defining the tax administration process. In this thesis, the Interaction Structure Diagram shown in Figure 2.2 is used to express the tree structure of the essential model. The Interaction Structure Diagram uses the products that correspond to the found transaction to present the overall structure of the process.

As can be seen from the diagram, the process of official decision creation consists of two sub-processes that are crucial for the decision creation. Additionally, five sub-processes can be performed based on each case's specific circumstances. Furthermore, three previously found transactions depend on third person participation in the proceedings. The necessity of each transaction is defined with a cardinality that can be seen near the association in the diagram. By default, the cardinality is 1 (the sub-process is mandatory), but its value can be specified to express the need for the corresponding sub-process. For example, the creation of the expert report is not necessary in each case, so its cardinality is defined as *0..1*, which means that the activity can be performed, but it is not crucial, and in some cases, the expert report is not created. In the same way, cardinality *0..** shows that for the determination of tax can be necessary the creation of multiple testimonies or participation of several persons.

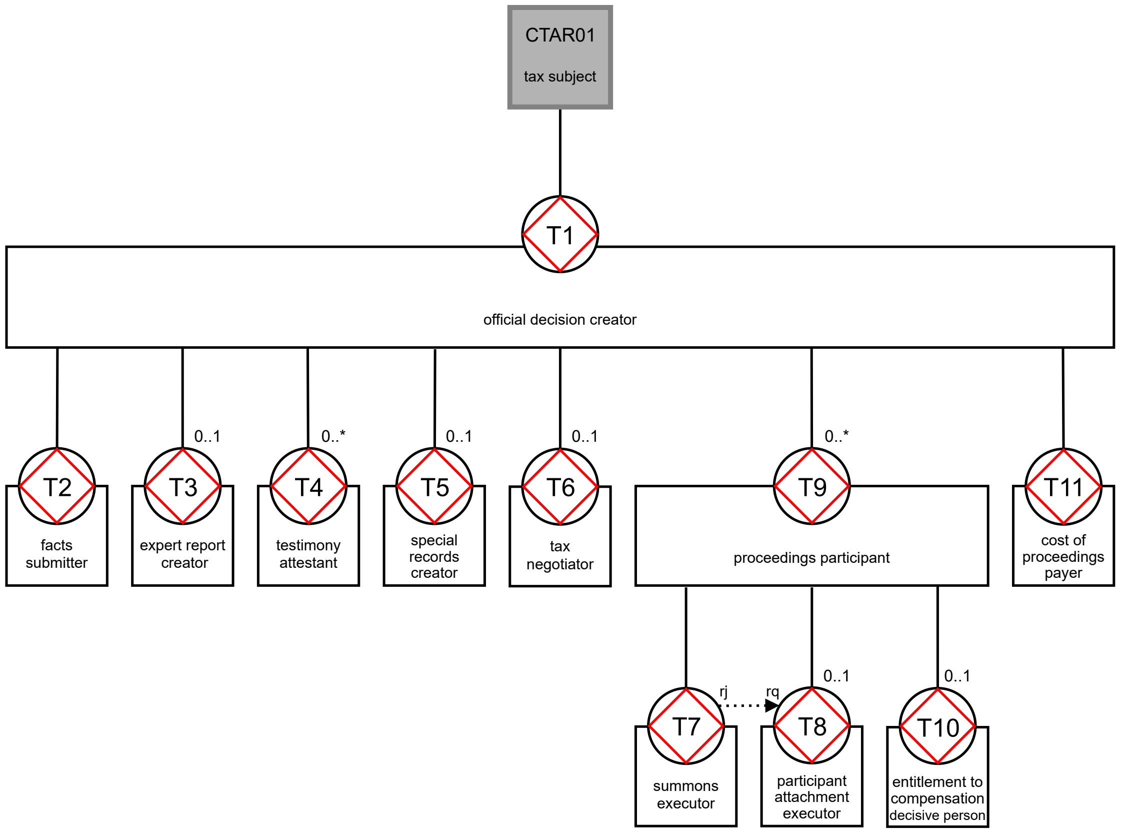


Figure 2.3 Coordination Structure Diagram of tax administration process.

2.2.4 Validating the essential model

As recommended by Dietz and Mulder [17], the final step of the OER analysis involves validating the previously performed steps. This step focuses on verifying the essential model and its transactor roles to ensure that the claims made by the model are accurate.

The validation procedure confirms that the construction of the essential model and other constructs with minor changes (which are already included in the diagram and tables presented in the previous sections to show the final correct results of OER analysis) comply with all validation rules that come from validation from ontology. This emphasizes that the essential model and constructs defined in the previous steps are valid and adhere to the techniques used to achieve essence and simplicity in DEMO.

2.3 Coordination Structure Diagram

The first created model, the Cooperation Model, is represented by a Coordination Structure Diagram. The construction of the model itself is based on the 11 transactions that were found during the OER analysis and were formerly presented in an Interaction Structure Diagram. The Coordination Structure Diagram of the tax administration is shown in Figure 2.3. The diagram displays all the found transactions as well as the overall coordination of the actors who are responsible for the completion of the associated activities.

The actors who perform the acts are named as recommended by Dietz and Mulder [17]. The actor role names correspond to the name of the transaction, of which the actor is the executor, and the final product of the transaction. The primary purpose of the naming is to avoid uncertain actor names or, even worse, naming actors after real-world persons. Due to the proposed standardized naming conventions, the actor's associated transaction and its product are clearly visible and expressive.

To summarize the presented model, the process itself starts with the tax subject proposing the official decision creation. The creation of the official decision, for which the official decision creator is responsible, consists of several crucial activities and some activities that are unnecessary in certain cases, as defined by the cardinality expressed in the same way as in the Interaction Structure Diagram case. The main part of the official decision creation corresponds to the submission of the facts based on which the tax is determined. As the facts may not be sufficient for tax determination, the official decision creator can initiate the processes of expert report or testimony creation, as well as the processes of tax negotiation or third person proceedings participation. At the end of the creation process, the payment of the proceedings cost is realized.

The third-person participation is initiated for the purpose of additional contribution beyond the information collected from other activities. To compel the participants to participate in the proceedings, they are firstly summoned. If they refuse the summons, the attachments of the participants are performed by the attachment executor, as defined by the arrow between the transactions (rejection of the summons initiates the request of attachment). If the participant loses a verifiable profit due to participation in the proceedings, he can ask for compensation. The decisive person determines the entitlement to compensation.

2.4 Object Fact Diagram

To visualize the products of the transaction, the Fact Model is used. The Object Fact Diagram shown in Figure 2.4 represents the Fact Model of the tax administration and emphasizes the data aspect of the process. In comparison to the Coordination Structure Diagram, which focuses on the overall structure of the process, the Object Fact Diagram provides an overview of the values and information that occur during the process execution, as well as the states of the entities involved.

The model includes two core entity types: *Official decision* and *Proceedings participant*. The connection between these types is established by the property type *the participant of [official decision] is [proceedings participant]*. Additionally, the model contains entity types corresponding to the actions required to determine the tax and create the official decision.

In addition to the element types, the code of law that describes the tax administration process implies additional attribute types. The element type *official decision* includes the attribute type *date of publication* and *tax amount*. The attribute types *negotiated tax amount*, *cost of proceedings amount*, and *compensation amount* are used to define the negotiated tax, the cost of the proceedings, and the compensation of the participant, respectively.

2.5 Other models

DEMO defines two additional types of models. The Action Model describes the manifestation of a construction, while the Process Model provides a more detailed definition of the effects of each act. However, since the Cooperation Model and Fact Model, along with the e-TRT of each transaction and Subject-Actor Table, are sufficient for creating a BPM system to support the tax administration process, these models are not included in this thesis.

Creating the Process Structure Diagram to represent the Process Model itself is not crucial because most transactions are initiated by the promise of the official decision creation or proceedings participation transactions. Moreover, the executions of the transactions representing the sub-processes of official decision creation are mutually independent. The only exception in the entire process involves the summons-attachment dependency, which has already been explained.

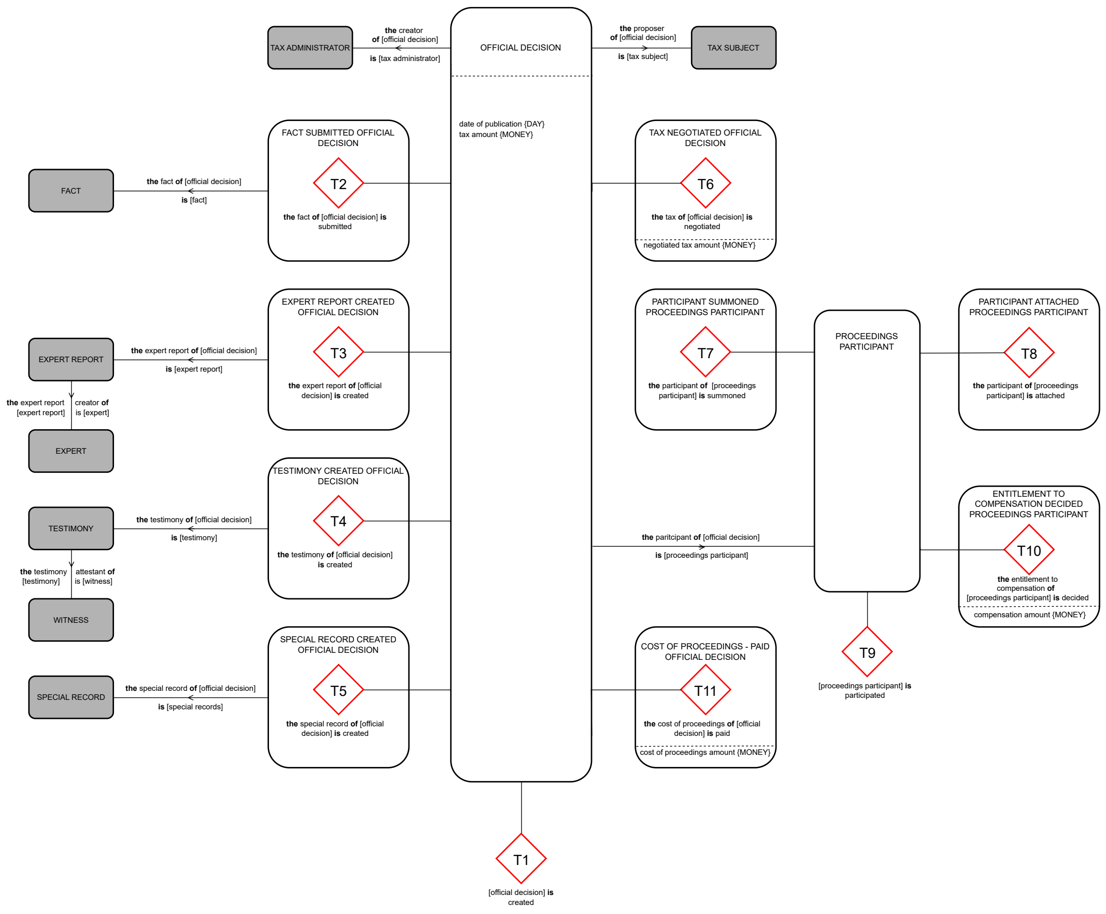

TAX ADMINISTRATOR
the creator of [official decision] is [tax administrator]
OFFICIAL DECISION
date of publication {DAY}
tax amount {MONEY}
the proposer of [official decision] is [tax subject]
TAX SUBJECT
FACT
the fact of [official decision] is [fact]
FACT SUBMITTED OFFICIAL DECISION
T2
the fact of [official decision] is submitted
EXPERT REPORT
the expert report of [official decision] is [expert report]
EXPERT REPORT CREATED OFFICIAL DECISION
T3
the expert report of [official decision] is created
the expert report [expert report]
creator of is [expert]
EXPERT
TESTIMONY
the testimony of [official decision] is [testimony]
TESTIMONY CREATED OFFICIAL DECISION
T4
the testimony of [official decision] is created
the testimony [testimony]
attestant of is [witness]
WITNESS
SPECIAL RECORD
the special record of [official decision] is [special records]
SPECIAL RECORD CREATED OFFICIAL DECISION
T5
the special record of [official decision] is created
TAX NEGOTIATED OFFICIAL DECISION
T6
the tax of [official decision] is negotiated
negotiated tax amount {MONEY}
PARTICIPANT SUMMONED PROCEEDINGS PARTICIPANT
T7
the participant of [proceedings participant] is summoned
PROCEEDINGS PARTICIPANT
PARTICIPANT ATTACHED PROCEEDINGS PARTICIPANT
T8
the participant of [proceedings participant] is attached
ENTITLEMENT TO COMPENSATION DECIDED PROCEEDINGS PARTICIPANT
T10
the entitlement to compensation of [proceedings participant] is decided
compensation amount {MONEY}
the paritcipant of [official decision] is [proceedings participant]
T9
[proceedings participant] is participated
COST OF PROCEEDINGS - PAID OFFICIAL DECISION
T11
the cost of proceedings of [official decision] is paid
cost of proceedings amount {MONEY}
T1
[official decision] is created

Figure 2.4 Object Fact Diagram of tax administration process.

2.6 Summary

The presented state of the as-is analysis succeeds in its purpose of providing a formal foundation for subsequent BPMN models and BPM system and extracting the fundamentals from the tax administration process definition. The result of the analysis consists of the Coordination Structure Diagram and Object Fact Diagram, which together contain substantial information, as well as other constructs that emerge from the OER analysis, such as actor definitions and detailed transaction descriptions. All of these models and constructs can be found in the repository corresponding to the presented case study, which is part of this thesis.

During the OER analysis of the tax administration process, a total of 11 transactions were identified. If the law was defined ideally from the DEMO perspective, which means that all coordination acts were specified, it would be possible to identify 110 described coordination acts in total. However, the vast majority of coordination acts are performed tacitly, in other words, they are performed implicitly or can be deduced from the presence or absence of other acts.

Out of the 110 coordination acts that could be specified, only 31 were explicitly defined. The definition of the remaining 79 coordination acts can usually be derived from the text or is absent completely, meaning that 72% of the coordination act definitions are missing. To summarize the specified and unspecified types of acts, Table 2.4 is presented.

The most specified types of coordination acts are declare and request, with only 27% and 36% of transactions missing their definition, respectively. The most persuasive argument for this is that declare and request are acts that shape the product of the transaction itself. In comparison, the promise act is specified in only 2 transactions, corresponding to 82% of transactions missing its definition. This is due to the large number of tacit declarations of the act. The execution of the promise act can, in most cases, be deduced from context.

Furthermore, revokes were specified in only a few cases. In total, any type of revoke of basic coordination act type was defined in 4 transactions. In other words, 91% of revoke definitions were missing from the description of transactions in the code of law.

	Specified	Not Specified	Missing information
Standard transaction pattern			
Request	7	4	36%
Promise	2	9	82%
Decline	3	8	73%
Declare	8	3	27%
Reject	4	7	64%
Accept	3	8	73%
Total	**27**	**39**	**59%**
Revokes			
Revoke request	2	9	82%
Revoke promise	1	10	91%
Revoke declare	1	10	91%
Revoke accept	0	11	100%
Total	**4**	**40**	**91%**
Complete transaction pattern			
Total	**31**	**79**	**72%**

Table 2.4 Missing transaction steps.

Chapter 3

Digitization to-be analysis

After describing the as-is analysis in the previous sections, this chapter will focus on the to-be analysis of the tax administration process defined in the Czech code of law. The goal of this chapter is to create and present a to-be state of the process, which can subsequently be used as a template for the creation of the final BPM system supporting the process that was analysed using DEMO.

3.1 Camunda Modeler

To capture the to-be state of the process, a BPMN level 2 to-be model needs to be created. In addition to the workflow engine and other components helping with orchestration, Camunda offers a modeler that can be used to create and edit the BPMN models. Currently, with the release of Platform 8, Camunda has introduced a new Modeler, and there are now two ways to create a BPMN model using Camunda products. A new Web Modeler has been added to the well-known and widely used Desktop Modeler in order to conform to the recent direction of Camunda, which is based on a Software as a service (SaaS) software distribution system, as described by Tavlasto [30].

The changes in comparison with the Desktop Modeler are mainly based on the added possibilities for collaboration, which consider sharing of the models or real-time collaborative model creation. Both the Web and the Desktop Modeler offer the same constructs that are used to create BPMN 2.0 models, which can further drive the process engine (meaning the possibility of creating the BPMN executable model), as defined in [31]. Since the Desktop Modeler covers the same amount of BPMN 2.0 elements that are used in process modeling and the case study of the thesis is based on the Self-Managed version of Camunda Platform 8 (an alternative to the SaaS version, more thoroughly described in the chapter *Implementation of proof-of-concept BPM system using Camunda Platform*), all of the further presented BPMN models are modeled using the desktop version of the Camunda Modeler.

3.2 To-be model

The to-be model, which is divided for better clarity into Figures 3.1 and 3.2, represents the to-be state of the tax administration process after the deployment of the BPM system. The foundation for the to-be model is the as-is analysis that uses DEMO described in the previous chapter. The activities are derived directly from the presented DEMO models or can be derived from the extended Transaction Result Tables. The model is compliant with well-established conventions that are based on the Czech code of law.

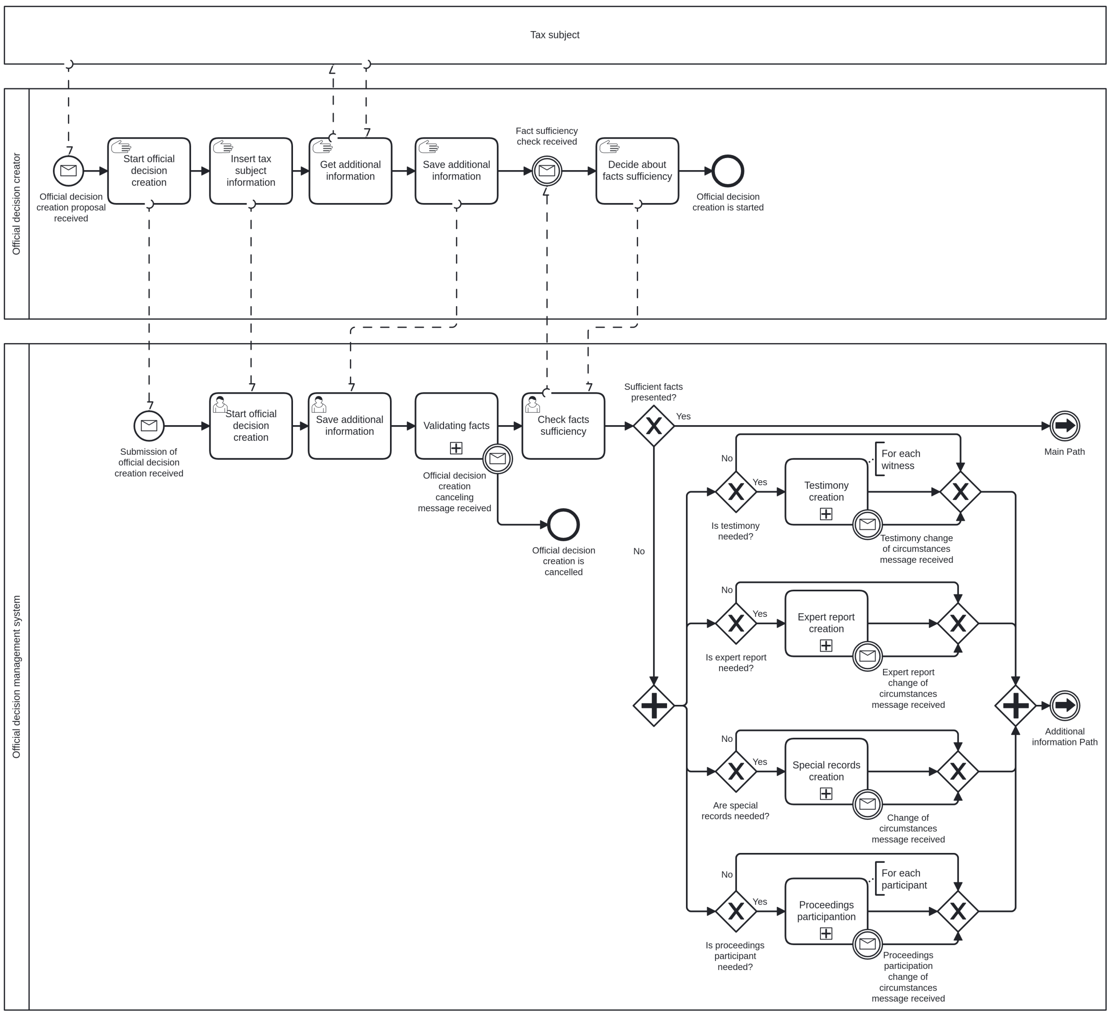

Tax subject
Official decision creator
Official decision management system
Official decision creation proposal received
Start official decision creation
Insert tax subject information
Get additional information
Save additional information
Fact sufficiency check received
Decide about facts sufficiency
Official decision creation is started
Submission of official decision creation received
Start official decision creation
Save additional information
Validating facts
Official decision creation canceling message received
Official decision creation is cancelled
Check facts sufficiency
Sufficient facts presented?
Yes
No
Main Path
Is testimony needed?
Testimony creation
For each witness
Testimony change of circumstances message received
Is expert report needed?
Expert report creation
Expert report change of circumstances message received
Are special records needed?
Special records creation
Change of circumstances message received
Is proceedings participant needed?
Proceedings participantion
For each participant
Proceedings participation change of circumstances message received
Additional information Path

Figure 3.1 Part 1 of analytical BPMN To-be model of tax administration process.

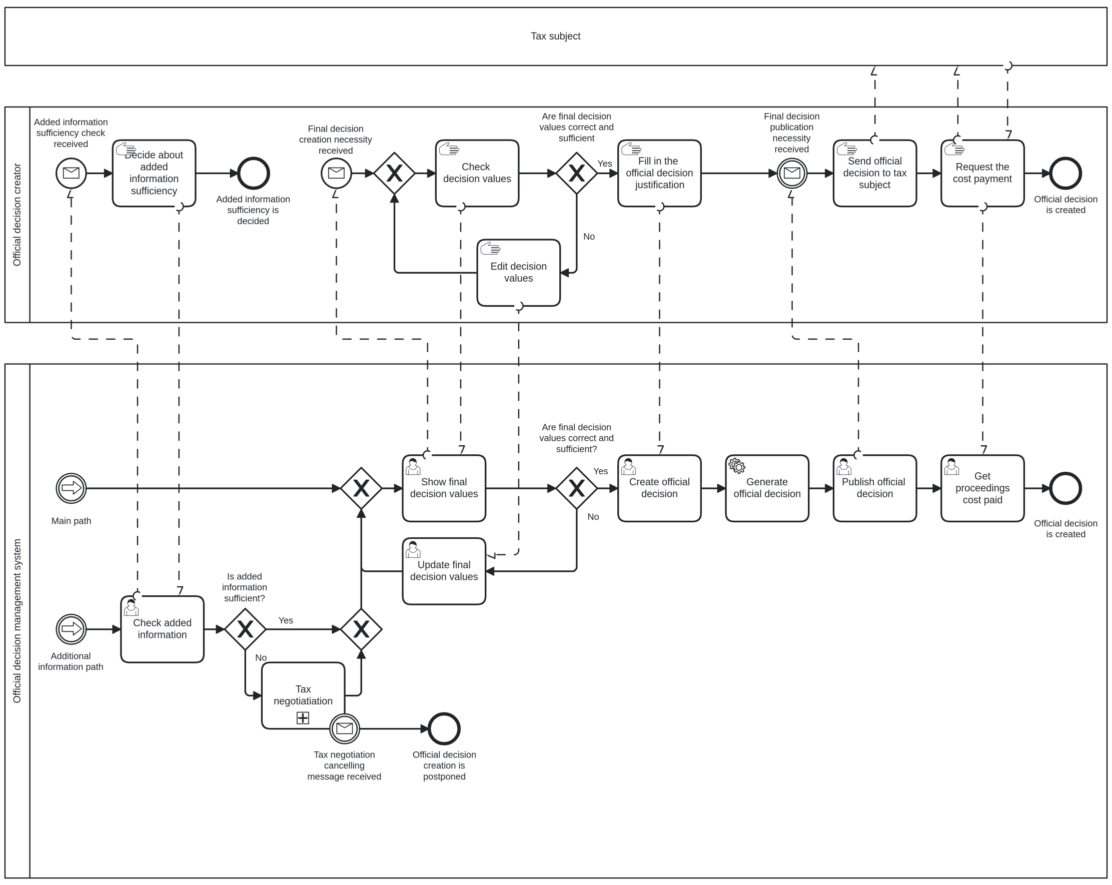

Tax subject
Official decision creator
Added information sufficiency check received
Decide about added information sufficiency
Added information sufficiency is decided
Final decision creation necessity received
Check decision values
Are final decision values correct and sufficient
Yes
No
Edit decision values
Fill in the official decision justification
Final decision publication necessity received
Send official decision to tax subject
Request the cost payment
Official decision is created
Official decision management system
Main path
Show final decision values
Are final decision values correct and sufficient?
Yes
No
Update final decision values
Create official decision
Generate official decision
Publish official decision
Get proceedings cost paid
Official decision is created
Additional information path
Check added information
Is added information sufficient?
Yes
No
Tax negotiatiation
Tax negotiation cancelling message received
Official decision creation is postponed

Figure 3.2 Part 2 of analytical BPMN To-be model of tax administration process.

3.2.1 Participants

The to-be BPMN model is constructed from the perspective of the management system. The main pool, named *Official decision management system*, represents the system itself. In addition to the management system pool, the to-be model or models of sub-processes contain 11 other pools that characterize the participants of the tax administration process. The participation of these participants during the execution of the process is or can be essential for the correct determination of the tax (as the as-is analysis showed and the to-be model implies, some of the actions are not necessary for the determination of the tax in specific cases, because of that the participation of the participants responsible for the actions is not obligatory). Some of the pools are collapsed as the process of these parties is not known in detail, and it is out of the scope of the modeled process. The participants that can occur during the tax administration process execution are:

- *Tax subject*
- *Official decision creator*
- *Testimony creator*
- *Witness*
- *Expert*
- *Special records supervisor*
- *Participant supervisor*
- *Summons executor*
- *Attachment executor*
- *Proceedings participant*
- *Tax negotiator*

Tax subject is the person whose official decision is created during the process execution. This participant is responsible for the initiation of the proceedings that officially begin with the sending of the proposal to create the official decision.

Official decision creator accomplishes most activities during the process execution. The participant mainly communicates with *Tax subject* to obtain the information necessary for the tax determination and performs tasks that correspond to the creation of the official decision. These activities include, for example, getting and validating facts presented by *Tax subject* or creating the final document. Aside from that, *Official decision creator* also decides on additional requirements that are crucial for the correct determination of the tax and the completion of the tax administration process, such as creation expert opinion or getting witness testimonies.

The responsibilities for completing additional requirements are distributed among multiple participants. *Testimony creator* is responsible for communicating with *Witnesses* to add necessary testimonies. Similarly, *Participant supervisor* collaborates with *Summons* and *Attachment executors* to obtain the contributions of *Proceedings participants* in the tax administration process. *Expert* has the responsibility for creating expert reports. Similarly, *Special records supervisor* should communicate with *Tax subject* to fulfill the need for special records creation. Lastly, *Tax negotiator* is responsible for tax negotiation if it is necessary for the correct process accomplishment.

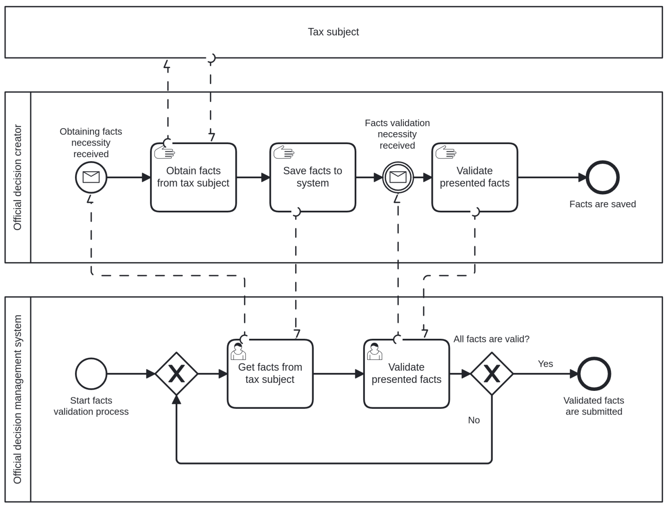

Tax subject
Official decision creator
Obtaining facts necessity received
Obtain facts from tax subject
Save facts to system
Facts validation necessity received
Validate presented facts
Facts are saved
Official decision management system
Start facts validation process
Get facts from tax subject
Validate presented facts
All facts are valid?
Yes
No
Validated facts are submitted

Figure 3.3 Validating facts sub-process.

Although the responsibilities for completion of diverse tasks are divided among multiple participants, in the final BPM system that will support the tax administration process, a real-world person can have multiple participant roles. For example, a tax administrator (a real-world person) has the main responsibility for creating the final decision. In addition to the obvious *Official decision creator* role, the roles *Testimony creator* or *Special records supervisor* can be assigned to the tax administrator, as he may be responsible for creation of testimony and obtaining of special records. The same applies to *Tax negotiator.* On the other hand, in the real-world, the persons responsible for the *Expert* and *Summons/Attachment executor* activities will differ in most cases from the persons responsible for the performance of the rest of the activities as they need to have the required education and powers, respectively. The partition of tasks between more participants with fewer activities is selected to achieve better intelligibility and clarity of the model from the perspective of participant responsibilities.

3.2.2 Process description

The reason why the BPMN model is presented is its ability to describe the orchestration of the activities required to execute the process. The whole *Official decision management system* consists of 10 tasks and 6 sub-processes that contain several additional tasks. Considering the tasks that are part of the sub-processes and the tasks that are placed in the pools representing participants described in the section before, the BPMN model consists of 58 tasks in total.

The order of the tax administration process activities is shown in BPMN models using the sequence flow element (solid line arrows), while the message flow (dashed line arrows) represents the path of messages between the pools.

As mentioned above, the process itself starts with the proposal to create the official decision. *Official decision creator* reacts to the proposal by initiating the process within the *Official decision management system* pool. After that, the initial information about *Tax subject* is submitted to the system. To register all the information that can be essential during the process, *Official decision creator* collects additional information from *Tax subject.*

Then follows the most important sub-process of the whole tax administration process that is focused on the collecting and validating presented facts. The *Validating facts* sub-process can be seen in Figure 3.3. Firstly, all the information that corresponds to the facts presented by *Tax subject* is written down and stored. Secondly, *Official decision creator* is able to go through the submitted information and check if everything is valid. In case the information is not valid, *Tax*

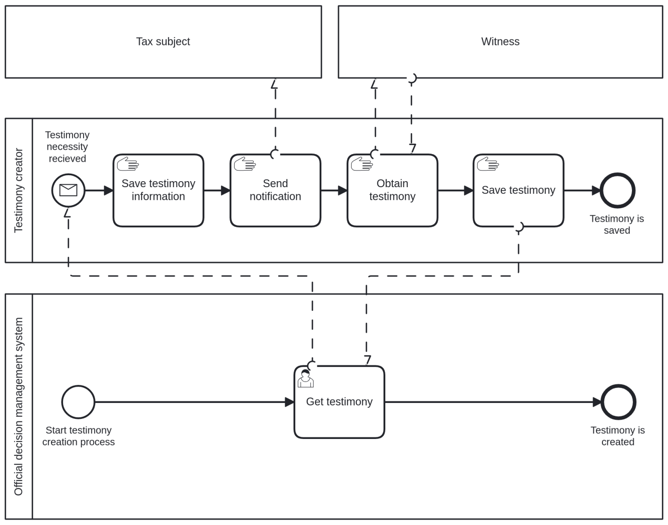


Figure 3.4 Testimony creation sub-process.

subject is requested to present the facts again in a correct way. If the validation of the presented facts does not reveal any flaw, the sub-process ends. During the sub-process, the canceling impulse that is typically based on the information gained during the process of obtaining facts can be given. This impulse is represented in the model as an interrupting message event.

Next, a check of the sufficiency of the presented facts for the creation of the official decision takes place. During the activity, *Official decision creator* decides whether the facts, together with additional information, are sufficient for the creation of the official decision or whether additional steps are crucial to complete the process.

If the *happy path* (kind of a default scenario with a typically positive outcome without deviations, exceptions, or errors as defined in [32]) is followed, the final check of the values that are crucial to the decision creation is completed before the official decision is created and published. The creation consists of determining the amount of tax that must be paid, together with the justification of the decision containing the reasons that lead to the determined amount of tax. With all the necessary information submitted and saved, the document representing the official decision is generated. Publication of the official decision is based on sending the signed decision to *Tax subject* and also saving it to the appropriate system.

The last activity of the whole process ensures that the proceedings costs are paid. *Official decision creator* interacts with *Tax subject* to pay the costs that occur during the activities corresponding to the determination of the tax and the creation of the official decision.

If the information arising from the presented facts is not sufficient for the tax determination, *Official decision creator* can opt to obtain the necessary additional information. In that case, witness testimonies, expert report, or special records must be created if required. Similarly, the acquisition of proceedings participation contribution can be added to the process. Each of the additional steps is modeled as a separate sub-process. These sub-processes are shown in Figures 3.4, 3.5, 3.6, and 3.7.

The sub-processes of *Testimony creation* and *Expert report creation* contain the activities that correspond to obtaining the testimony and creating the expert report, respectively. In the case of testimony creation, saving information about the witness and, more importantly, notifying *Tax subject* about the intention to create the testimony must precede, as revealed by the as-is analysis.

The sub-process of *Special records creation* consists of multiple activities. Firstly, *Tax subject* needs to be notified about the obligation to create special records. The obligation attributes are

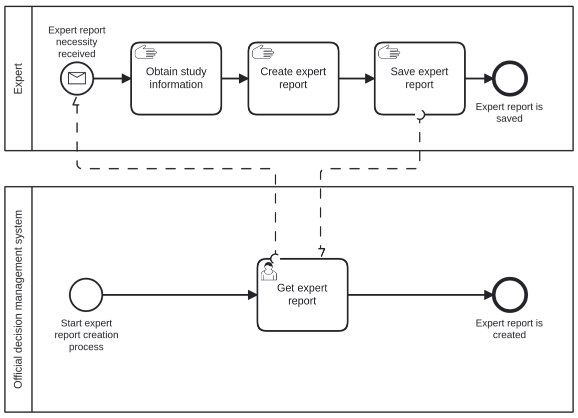


Figure 3.5 Expert report creation sub-process.

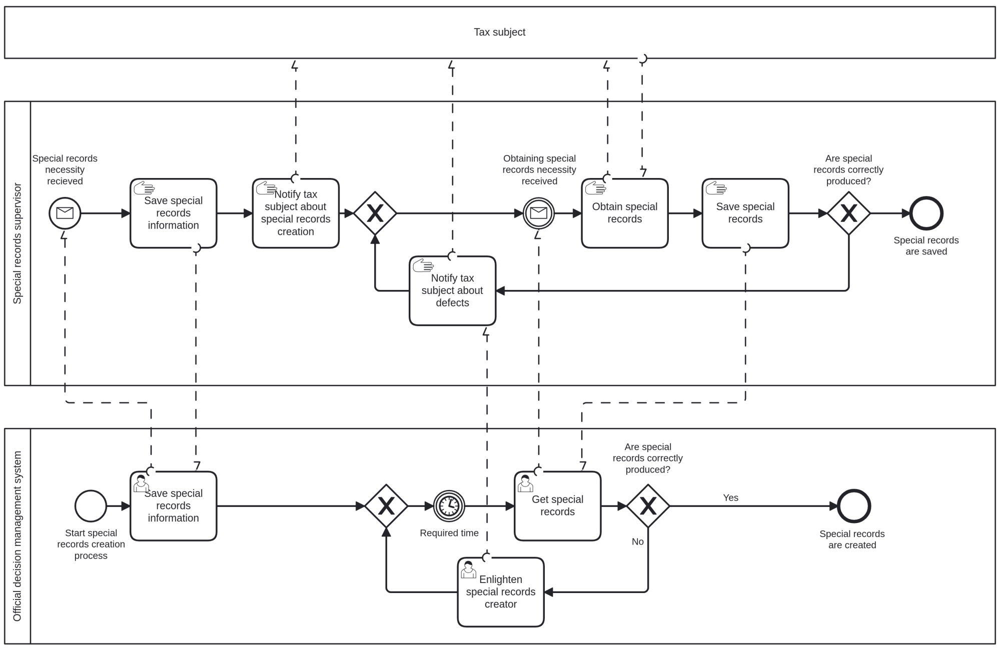

Tax subject
Special records supervisor
Special records necessity recieved
Save special records information
Notify tax subject about special records creation
Obtaining special records necessity received
Obtain special records
Save special records
Are special records correctly produced?
Special records are saved
Notify tax subject about defects
Official decision management system
Start special records creation process
Save special records information
Required time
Get special records
Are special records correctly produced?
Yes
No
Special records are created
Enlighten special records creator

Figure 3.6 Special records creation sub-process.

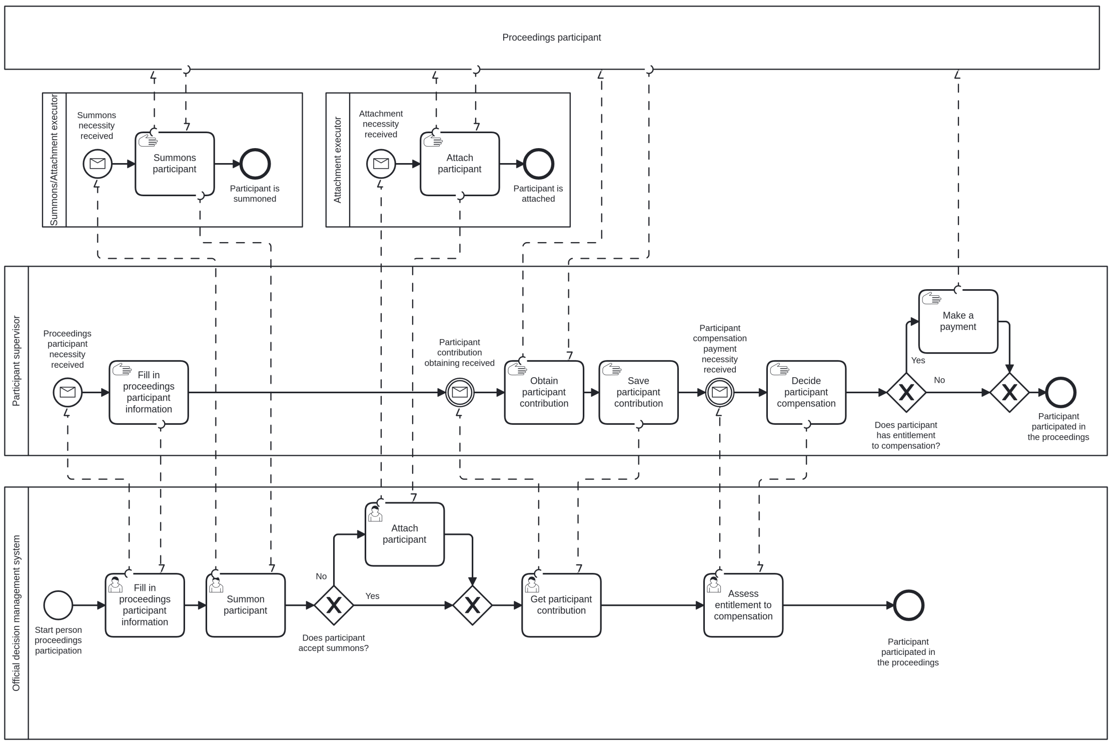

Proceedings participant
Summons/Attachment executor
Summons necessity received
Summons participant
Participant is summoned
Attachment executor
Attachment necessity received
Attach participant
Participant is attached
Participant supervisor
Proceedings participant necessity received
Fill in proceedings participant information
Participant contribution obtaining received
Obtain participant contribution
Save participant contribution
Participant compensation payment necessity received
Decide participant compensation
Yes
No
Does participant has entitlement to compensation?
Make a payment
Participant participated in the proceedings
Official decision management system
Start person proceedings participation
Fill in proceedings participant information
Summon participant
No
Yes
Does participant accept summons?
Attach participant
Get participant contribution
Assess entitlement to compensation
Participant participated in the proceedings

Figure 3.7 Proceedings participation sub-process.

based on information that has been saved before. After a required time (the time is not specified in the code of law, as it can vary in different cases), *Special records supervisor* collects the special records and decides whether the information collected is valid and whether the structure and content comply with the requirements. If defects or a lack of information are found, *Special records supervisor* enlightens *Tax subject* about the deficiencies and imposes the obligation to create special records for a specified period of time again.

To accomplish the *Proceedings participation* sub-process, firstly the information about *Proceedings participant* needs to be filled in. Later, the corresponding person who has the powers summons the participant. In case the summons is not accepted, the participant attachment takes place. The next step is based on obtaining and saving the participant's contribution, which can vary in different cases. The contribution of participant can be anything that can help with tax determination. The final steps consist of assessing the entitlement to compensation for participation and eventually its payment.

Since circumstances can change during the execution of these sub-processes, the gained information can become unnecessary. In this case, waiting for sub-process completion is delaying. That is the reason why the intermediate interrupting boundary event is attached to the sub-processes. The event ensures the termination of the sub-process once the message that corresponds to a change of circumstances is caught.

Following the accomplishment of the additional sub-processes (or its possible interruption), *Official decision creator* decides whether the current information is sufficient to determine the tax and create the official decision. If the information is insufficient, the final tax amount must be negotiated. *Tax negotiation* visible in Figure 3.8 is the last tax administration sub-process. The negotiation itself consists of one activity – *Get negotiated tax*. During the sub-process, the final amount of tax is negotiated together with *Tax subject*. In case the *Tax negotiation* sub-process is canceled, the official decision creation is postponed.

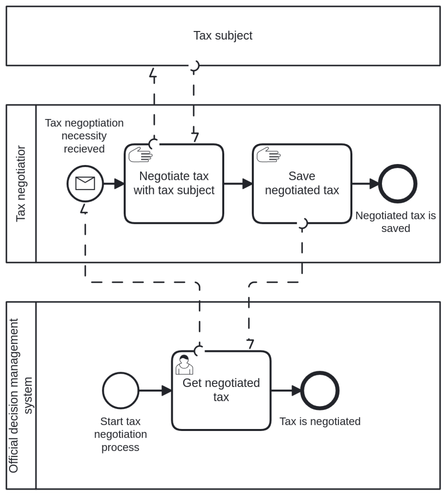

Tax subject
Tax negotiatior
Tax negoptiation necessity recieved
Negotiate tax with tax subject
Save negotiated tax
Negotiated tax is saved
Official decision management system
Start tax negotiation process
Get negotiated tax
Tax is negotiated

Figure 3.8 Tax negotiation sub-process.

3.3 Summary

The result of the to-be analysis of the tax administration process represents a proposed solution that can be used to implement a management system supporting the process. The proposed to-be state, visualized using a BPMN level 2 model, corresponds to the state of process after the completion of digitization.

Most of the activities, as well as the participants, are directly derived from the materials created during the as-is analysis. DEMO analysis provided a solid formal foundation for further BPMN modeling and significantly helped during the process of model creation.

However, the law modeling differs significantly from the modeling of standard business processes. Although the tax administration process can be roughly considered a business process, the need to comply precisely with legislation makes the modeling process notably more demanding. Moreover, thorough digitization would require simultaneous change approvals of the corresponding part of the code of law.

Even though the current state of Czech legislation does not allow the creation of a BPM system consisting mainly of automated activities, the proposed system can be used to orchestrate the activities and people responsible for their accomplishment during the process execution.

Chapter 4

Implementation of proof-of-concept BPM system using Camunda Platform

This chapter will focus on the implementation of a proof-of-concept BPM system based on the Camunda Platform 8, which supports the tax administration process as designed in the previous chapter. Firstly, the hosting possibilities of Camunda Platform 8 offered by Camunda will be covered. Next, the architecture of the system, including all its components, will be described. The creation of a Logical data model, an executable BPMN model, as well as the definition of Camunda Forms and all implementation details necessary for successful deployment and execution, will be presented. In addition to the executable model, part of the implementation is based on the Camunda client, which contains job workers. Finally, testing will be described and an overview of the final BPM system will be provided in order to clarify the ways in which it can be used.

The entire implementation of the proof-of-concept BPM system can be accessed in the repository corresponding to the thesis.

4.1 Camunda hosting

With the release of Platform 8, Camunda offers two versions of hosting that can be used to deploy and run the modeled process. Camunda Platform 8 is offered as a Software as a service (SaaS) or a Self-Managed solution, as described in [33]. The difference, as the names imply, is based on the way in which the application is run.

4.1.1 Software as a service

With the SaaS version, Camunda is responsible for hosting, and the care about the technical setup is shifted from the customer who wants to create the BPM system based on Camunda's solution. The main benefits of the Software as a service solution correspond to the fact that the installation and technical setup of the application are provided by Camunda. Due to this, the customer can fully focus on model creation and task and microservice orchestration, as mentioned in [33].

4.1.2 Self-Managed

As an alternative to using Camunda Platform 8 through SaaS, Camunda offers Self-Managed version that is hosted by the customer. The creation of the overall process automation solution and the work with Camunda Platform 8 are similar to the Software as a service version, as written in [34], but the system in the case of Self-Managed version runs on the customer's infrastructure, providing a better overall overview during the development process and production deployment. However, since the behavior of the system is entirely managed by the customer, installation, setup, performance, security, uptime, redundancy, and resource allocation must be considered, as described by Levy [35].

Camunda offers the following options to run Platform 8 in a Self-Managed fashion, as defined in [36]:

- Helm/Kubernetes
- Docker
- Manual

According to [36], Kubernetes and Docker are recommended options to run Camunda Platform 8 Self-Managed in production use. Additionally, the Docker Compose configuration is provided to enable running Camunda Platform 8 on a developer machine, although it is not optimized for production usage.

The manual option involves running the Java application on a local or virtual machine that must provide a supported Java Virtual Machine (JVM). Even though this approach offers significant flexibility by running the Camunda platform on a virtual machine or "*bare metal*", Camunda does not recommend it due to the detailed and complex configuration required to ensure correct interaction between components, as described in [36].

The Docker option has been selected for further development of the BPM system because it allows the creation of the proof-of-concept solution that is close to the production without the need of complex configuration and because Camunda provides a Docker Compose configuration for local development. However, the solution presented further in the thesis can also be run with minimal adjustments using Kubernetes or Manual options, or even the SaaS version of Camunda Platform 8, as the underlying principles and most of the implementation details remain the same, as written in [34].

4.2 Architecture

The entire Camunda Platform 8 is composed of several components, which can be considered mutually communicating services. These components are offered to achieve the orchestration, observation, and analysis of microservices together with human tasks. The overall architecture of Camunda Platform 8 Self-Managed is shown in Figure 4.1. Apart from the Camunda Modeler, which was already described in the previous chapter, the documentation of the Camunda Platform 8, specifically the Self-Managed version [34], defines the following components:

- Zeebe workflow and decision engine
- Identity
- Operate
- Tasklist
- Optimize
- Connectors

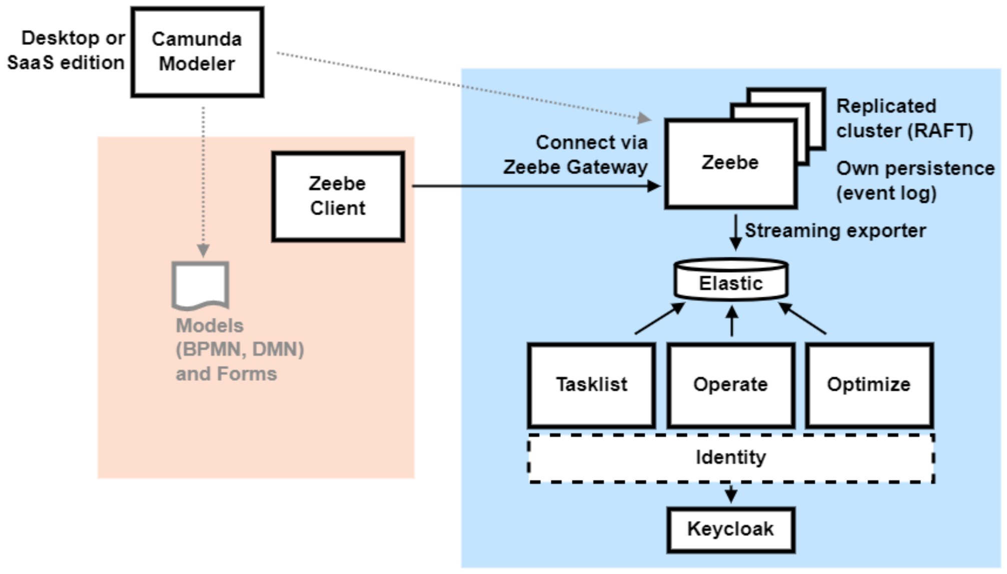


Figure 4.1 Architecture of Camunda Platform 8 Self-Managed [37].

4.2.1 Zeebe engine

Zeebe is the BPMN workflow engine that powers Camunda Platform 8, as defined in [38]. In order to ensure resilience, Zeebe is based on a fail-over architecture that also supports geo-replication across data centers to provide high availability. The architecture of Zeebe itself is visible in Figure 4.2. According to [39], the Zeebe engine is based on four main parts:

- Brokers
- Gateways
- Exporters
- Clients

The Zeebe broker is the distributed workflow engine that tracks the state of active instances of processes. The broker does not include any business logic. Its only responsibilities correspond to processing commands that have been sent by clients, storing and managing the state of active process instances, and assigning jobs to job workers, as described in [39]. Multiple Zeebe brokers, which form a peer-to-peer network in order to eliminate a single point of failure, create Zeebe cluster.

The gateways serve as an entry point to the Zeebe cluster. Their goal is to forward requests to brokers. In the Zeebe workflow engine, the gateways are stateless and sessionless. To meet the needs of load balancing and high availability, the gateways can be added and appropriately configured.

According to the Zeebe documentation [39], "*the exporter system provides an event stream of state changes within Zeebe.*" The purpose of the streamed data can vary, however, in most cases, it is used to monitor the current state of the process, track incidents, and analyse historical process data to optimize the process in the future. To provide data streaming, Zeebe includes an out-of-the-box Elasticsearch exporter.

Elasticsearch is a distributed search and analytics engine for various types of data, as defined in [40]. It is used as a tool to store imported historical data, as well as reports, dashboards,

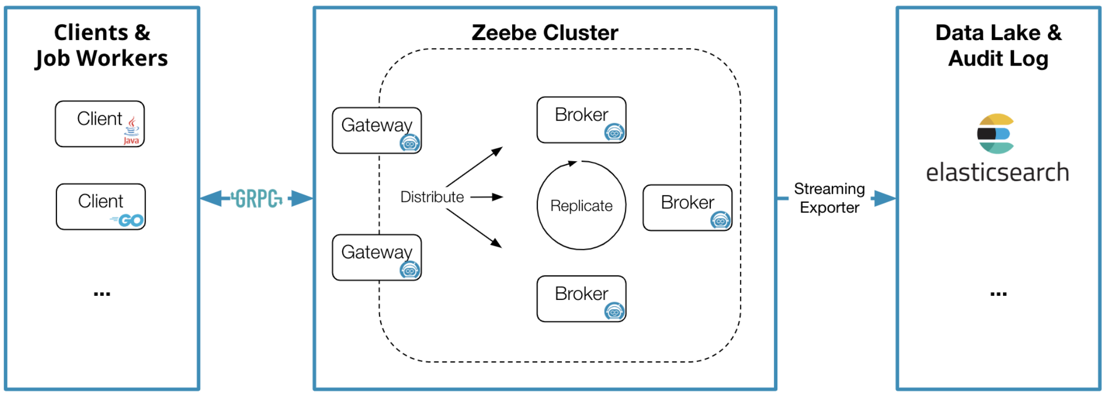


Figure 4.2 Architecture of Zeebe [39].

or alerts. Furthermore, the Self-Managed version also includes a Kibana profile in the Docker Compose file. According to [41], Kibana is a tool that can be used to search and visualize the data stored in Elasticsearch.

Since the gateways, brokers, and exporters are pre-configured in Camunda Platform 8 to provide the service, custom actions and additional implementation are mainly based on the clients. The client is used to send commands to Zeebe to deploy the process, perform business logic, or handle operational issues. In other words, the client is a microservice that connects to the Zeebe cluster to activate, complete, or fail jobs, as the brokers do not execute any business logic, as referred to in [39].

The connection between clients and Zeebe gateways is done using gRPC, which utilizes the HTTP/2-based transport protocol. Although Zeebe officially includes support for Java and Go clients, gRPC makes it possible to create clients in other programming languages as well, as described in [42].

4.2.2 Identity

Identity is the component within Camunda Platform 8 Self-Managed that is responsible for managing and ensuring authorization and authentication, as defined in [43]. The Identity component is used to recognize different users based on their user accounts and determine the actions they are permitted to perform.

The Identity component allows the management of applications, APIs, permissions, and roles. Using the permissions assigned to the applications, access to the components can be controlled. APIs refer to a service that provides various resources. Roles represent a way to group sets of permissions. Subsequently, these roles are assigned to real-world users to define their permissions as described in [43]. To configure the roles and APIs and manage permissions, Camunda Platform 8 Self-Managed offers an Identity UI.

Principles of Identity+ are based on Keycloak, a single sign-on solution used for web applications and RESTful web services, as described in [44]. In Camunda Platform 8, it is used as an identity and access management solution to secure access to other components such as Tasklist, Operate, and Optimize.

4.2.3 Operate

Operate is a tool used in Camunda Platform 8 to monitor and troubleshoot process instances running on the Zeebe Engine. In addition to providing visibility into running and completed process instances, the Operate component also offers the possibility to resolve incidents that occur during process execution or update process instance variables using built-in operations, as written in [45].

Using the Operate component, process instances can be easily retried or canceled if required, for example, due to underlying problems or incidents. These operations can also be performed for multiple process instances at once using the specified selections. The particular problems that occurred during process execution can be solved using the Operate component without the need to cancel or retry the execution of the instance. Errors during execution can occur due to erroneous and missing variable values, which can be resolved by editing and adding variables, respectively, as described in [45].

4.2.4 Tasklist

Camunda Platform 8 offers Tasklist to orchestrate tasks that require manual work to be completed. Tasklist is an application that provides a solution to processes containing user tasks in Zeebe and controls the human workflow crucial for the execution of the process through an interface for manual work, as defined in [46].

Tasklist is based on user notification principles. When a process is executed, the Tasklist system notifies the user of the need to complete a user task if the user is assigned to it. Additionally, the user can easily claim the task when working on its completion, indicating to other users that they can focus on other available tasks. This provides flexibility in managing multiple process instances while also eliminating the possibility of users being unaware of the need to complete tasks, as well as avoiding multiple users working redundantly on the same task, which can lead to prolonged process execution.

To complete the task, the system requires interaction with the user after they claim the task. In most cases, this interaction involves filling out the Camunda Form, updating or adding variables, or simply completing the task by marking it as done if the action is fully independent of the system. If a user is unable to complete the task, it can be unclaimed so that anyone else with the required skills or knowledge can claim it.

4.2.5 Optimize

To thoroughly and effectively complete process automation, the following three key aspects must be considered, as defined in [47]:

- Design
- Automate
- Improve

To design and automate the process, the components described above, such as Tasklist and Optimize, are used. Because Camunda Platform 8 is built to handle all three aspects of process automation, it contains the Optimize component, which is used to leverage process data gained during executions. Optimize offers business intelligence tools for users to collaboratively access reports, find bottlenecks, and examine areas for process improvements, as described in [47].

Reports and dashboards created with Optimize consist of several actionable insights that help discover improvement possibilities and find areas responsible for process execution delays. Both reports and dashboards can be assembled from different summarizing parts, such as the numerically expressed incident rate, the number of completed executions, or even heat maps based on the average distribution of total duration between process activities.

4.2.6 Connectors

According to the documentation [48], Connectors are reusable building blocks that perform integration with external systems. They are represented as separate tasks in the BPMN process model and can be configured with parameters specific to the external system. Using a Connector can eliminate the need to write additional code to integrate the external system.

Camunda Platform 8 (available in both SaaS and Self-Managed version) offers out-of-the-box Connectors that can be used for various tasks. For example, the SendGrid Connector enables sending emails directly from the BPMN process, the Google Drive Connector can be used to create folders or files from a Google Drive template, the REST Connector allows for easy requests to REST APIs and using the response in the following step of the process, and the Kafka Producer Connector can produce a message to Kafka from the BPMN process task. In addition to these, another 10 out-of-the-box Connectors are currently available in Camunda Platform 8 as referred to in [49].

Aside from the out-of-box Connectors, custom Connectors can also be developed when using the Self-Managed version of Camunda Platform 8. The implementation of custom Connectors is based on the Connector SDK, which allows for development in Java with already provided APIs for common Connector operations, and the Connector Template, represented by a JSON configuration file defining the appearance of the BPMN element and the ways in which it can be further configured.

4.3 Logical data model

To provide a conceptual representation of the data used in the system, the Logical data model is created. According to [50], it helps to define the structure and relationships between the data elements and provides a clear understanding of the meaning and important rules associated with the data.

The resulting Logical data model, which represents the data that occur during the process execution, is shown in Figure 4.3. The elements shown in the Logical data model are mostly based on the as-is analysis presented in one of the previous chapters, as they are directly stated in or arise from the Object Fact Diagram.

The main element of the model is *Official decision*, whose creation is the goal of the whole process. Data elements related to *Official decision* include *Tax subject*, *Presented facts*, *Cost* (representing the costs of the proceedings that correspond to tax administration), and data elements based on additional information such as *Testimony*, *Expert report*, *Special records*, and *Participant contribution*. The parties to the proceedings must have defined their *Addresses*, and the participant, as mentioned in the previous chapters, can have the right to compensation represented by the *Compensation* component in the model.

4.4 Executable BPMN model

The main part of the implementation of the BPM system is formed by the executable BPMN model, which is mainly based on the analytical BPMN model presented in the previous chapter. This model represents the backbone of the entire implementation, as the additional implementation details are mostly attached directly to its activities, gateways, or flow elements. Along with the necessary configuration and all the implementation details, the executable model can be deployed to the Zeebe process engine.

As can be seen in Figures 4.4, 4.5, and 4.6, which contain the executable BPMN model of the tax administration process (divided into three parts for better clarity), the overall activities and their orchestration are similar to the analytical model. However, the executable model is focused exclusively on *Official decision management system*.

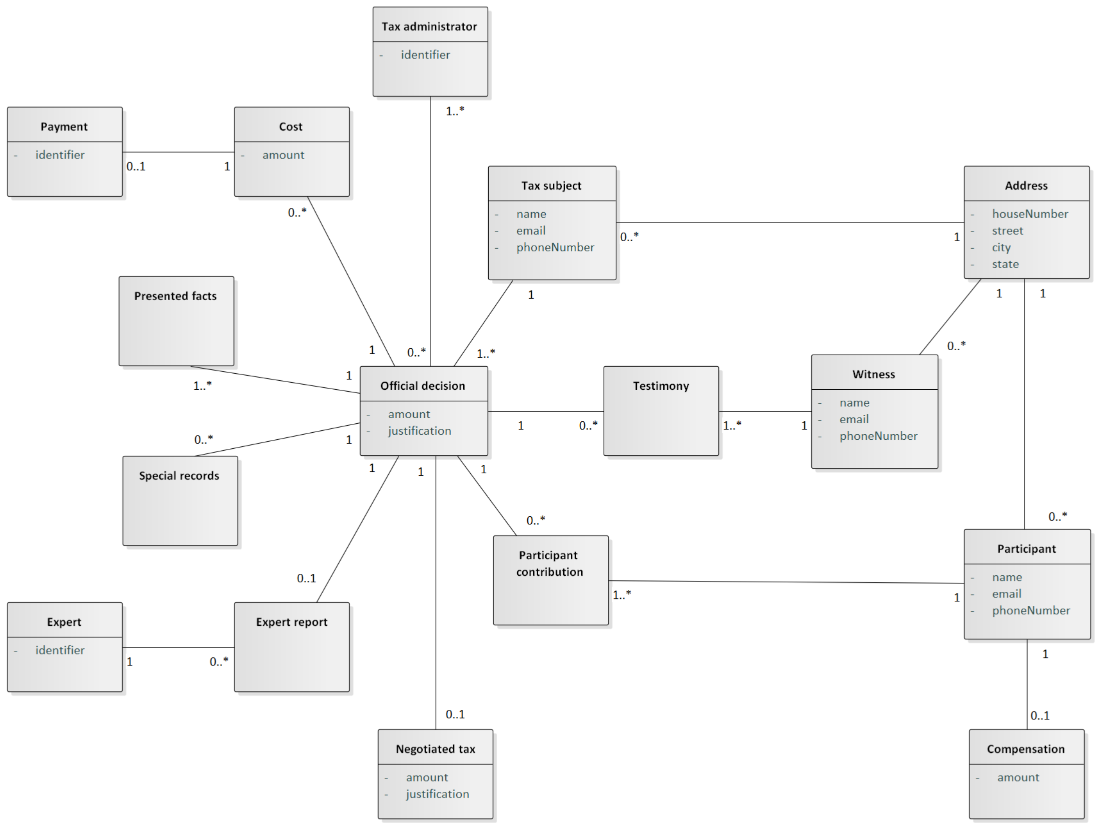


Figure 4.3 UML Logical data model of tax administration.

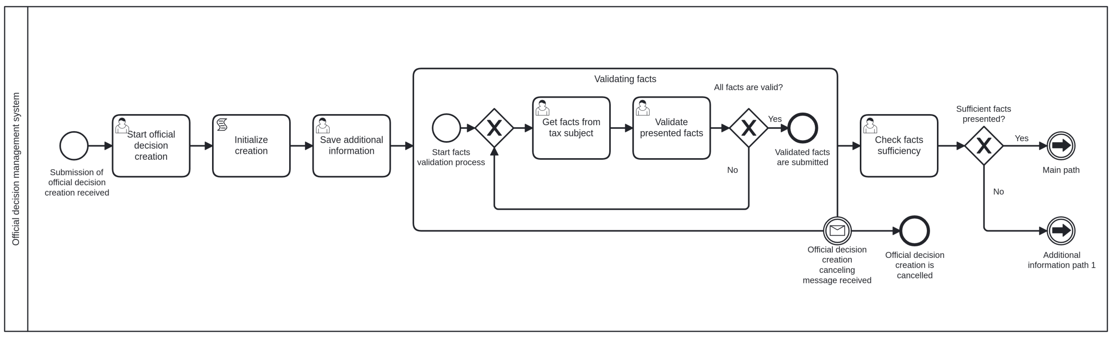

Official decision management system
Submission of official decision creation received
Start official decision creation
Initialize creation
Save additional information
Validating facts
Start facts validation process
Get facts from tax subject
Validate presented facts
All facts are valid?
Yes
No
Validated facts are submitted
Official decision creation canceling message received
Official decision creation is cancelled
Check facts sufficiency
Sufficient facts presented?
Yes
No
Main path
Additional information path 1

Figure 4.4 Part 1 of executable BPMN model of tax administration process.

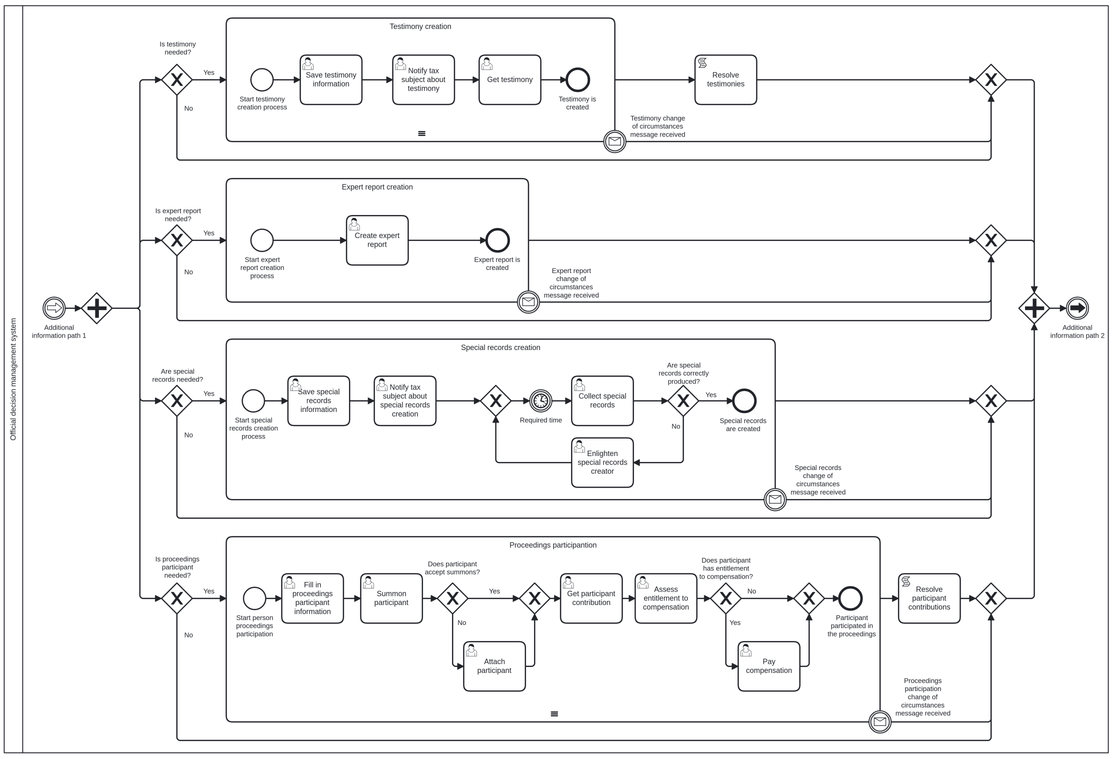

Official decision management system
Additional information path 1
Is testimony needed?
Yes
No
Testimony creation
Start testimony creation process
Save testimony information
Notify tax subject about testimony
Get testimony
Testimony is created
Testimony change of circumstances message received
Resolve testimonies
Is expert report needed?
Expert report creation
Start expert report creation process
Create expert report
Expert report is created
Expert report change of circumstances message received
Additional information path 2
Are special records needed?
Special records creation
Start special records creation process
Save special records information
Notify tax subject about special records creation
Required time
Collect special records
Are special records correctly produced?
Enlighten special records creator
Special records are created
Special records change of circumstances message received
Is proceedings participant needed?
Proceedings participantion
Start person proceedings participation
Fill in proceedings participant information
Summon participant
Does participant accept summons?
Attach participant
Get participant contribution
Assess entitlement to compensation
Does participant has entitlement to compensation?
Pay compensation
Participant participated in the proceedings
Resolve participant contributions
Proceedings participation change of circumstances message received

Figure 4.5 Part 2 of executable BPMN model of tax administration process.

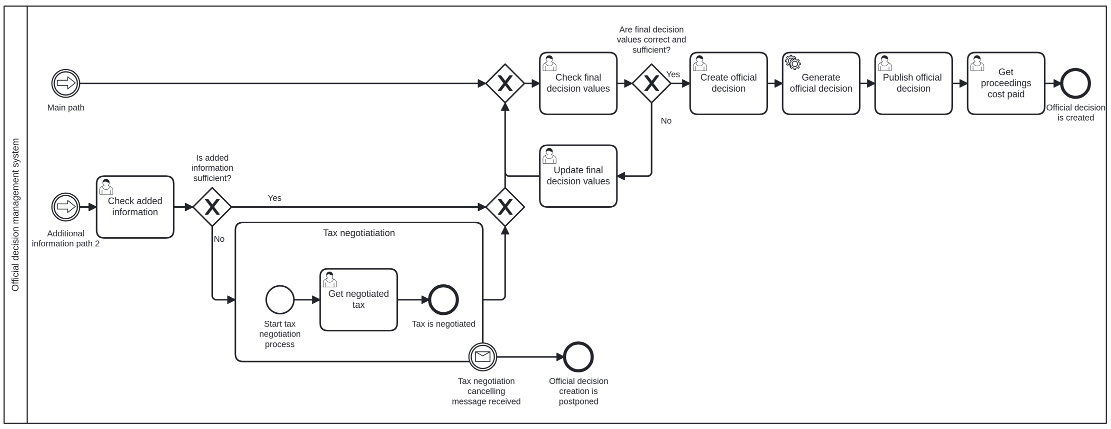

Official decision management system
Main path
Additional information path 2
Check added information
Is added information sufficient?
Yes
No
Tax negotiatiation
Start tax negotiation process
Get negotiated tax
Tax is negotiated
Tax negotiation cancelling message received
Official decision creation is postponed
Check final decision values
Are final decision values correct and sufficient?
Update final decision values
Create official decision
Generate official decision
Publish official decision
Get proceedings cost paid
Official decision is created

Figure 4.6 Part 3 of executable BPMN model of tax administration process.

The resulting process model mainly contains user tasks, as human involvement is necessary during most of the activities. The main benefit of the BPM system in this case is based on the advanced orchestration of human activities that are required to be executed for the successful completion of the tax administration process. This enables simplified communication between people during the official decision creation process.

The executable model contains additional tasks compared to the analytical model that are essential for implementation, such as the *Initialize creation* task, responsible for the initialization of the process variables and generating the unique identifier of the process. Other tasks that occur newly in the executable model have similar purposes.

The most significant changes are not visible from the model itself. These changes are based on implementation details that are typically hidden in the XML specification of the model. To deploy and correctly run the modeled process using the workflow engine, such as that offered by Camunda, these details need to be specified.

4.4.1 Forms description

One of the implementation details that is not directly visible from the executable BPMN model is the specification of Camunda Forms. As defined in [51], Camunda Forms are used to design and configure forms that can be connected to a user task to implement a task form in the final BPM system. After deploying the model containing configured Camunda Forms, Tasklist imports the form schema and uses it to render the form of every task assigned to it.

Camunda Forms, as well as the BPMN executable model, can be designed and configured using the Camunda Modeler. Once the design and configuration are done, the created form, which is in the case of the Desktop Modeler represented by JSON, can be assigned to one or more user tasks.

The design process of the Camunda Form consists of the composition of required Form Elements. According to the documentation [52], the following Form Elements are currently available in Camunda Platform 8:

- Text view
- Text field
- Text area [1]
- Number field
- Datetime [1]
- Checkbox
- Radio
- Select
- Checklist [1]
- Taglist [1]
- Image view [1]
- Button

[1] Supported by newest version of Camunda Platform – Camunda Platform 8.2, the elements are not available in older versions.

The configuration part is typically based on the specification of the behavior, which is defined using the properties of each Form Element, as well as the definition of data entry, since most of the Form Elements are intended to be bound to a variable using a data binding. To bind the variable to the Form Field (Form Element that can be bound to a process variable), the *Key* attribute is used. This attribute serves as an identifier to map the data of the respective field during the initial loading and also during the submission of the form, as described in [53].

Aside from data binding, the Form Element definition can optionally contain other additional attributes. These attributes are used to configure if the user can edit the element, to describe the required input (email, phone number, or eventually custom expression represented by regex value), and to define if the element must be filled before submission of the form or if it is possible to leave it empty.

Because the newest version of Camunda Platform 8 - version 8.2 - offers the option to create dynamic forms, the visibility of the Form Elements can differ over time. It is possible to leave the Form Element visible or eventually hide it. This enables the creation of forms that can change their structure in real-time based on the filled information. For example, in the presented solution, the form that corresponds to the *Check facts sufficiency* task is dynamically updated if the user marks the *Required testimony* checkbox, as he indicates the necessity to create testimony. In this case, the new text field is shown to enter the names of witnesses that are used later in the process execution.

4.4.2 Variables

Variables represent the data of the process instance. Each variable has a name that serves as an identifier and a JSON value where the data are stored. Camunda Platform 8 supports String, Number (which can be an integer or decimal number), Boolean, Array, Object, or Null data types as described in [54].

To understand working with variables, it is necessary to know the basics of variable scopes and propagation. The visibility of a variable is defined by its variable scope. The root scope is the process instance itself. When a variable is in the root scope, it is visible and accessible anywhere in the process. When a process enters a sub-process, a new scope is created. The activities in this scope can see any variables of this and the parent scopes. On the other hand, activities outside the scope cannot access the variables defined there. If the names of the variables outside and inside the scope are the same, the parent scope variable is temporarily covered. In this case, activities in the child scope cannot access the value of the variable in the parent scope.

In case a variable is merged into a process instance (which happens on job completion or on message correlation), the variable is propagated from the scope of the activity to higher scopes. The propagation ends when a scope defines a variable with the same name whose value is updated. If no scope contains a variable with the same name, a new variable is created in the root scope. The propagation can be limited using local variables. When a variable is set as a local variable, it is created or updated in the given scope, regardless of the variable with the same name in the parent scopes [54].

To create a new variable or customize how variables are merged into a process instance, the input and output variable mapping can be used. In the process, the mapping is defined as an extension element under *ioMapping*. Each variable mapping contains a *source* and *target* expression. The *source* expression defines the value of the mapping. In most cases, that means accessing the instance variable that holds the desired value. On the other hand, the *target* expression describes where the mapped variable should be stored. As the variable mappings are evaluated in the defined order, the *source* expression can access the *target* expressions of previous mappings. If one or more output mappings are used, other variables are set as local variables in the corresponding scope.

To access the variables and eventually calculate the values dynamically, expressions are used. In Camunda, the expressions are written in FEEL (Friendly Enough Expression Language), which is part of the OMG's DMN specification. Camunda Platform 8 integrates the FEEL Scala engine to evaluate expressions, as referred to in [55]. Expressions are required in some of the attributes of BPMN elements. For example, the condition of a sequence flow on an exclusive gateway, as well as input and output collections of a multi-instance activity, must contain FEEL expression. Aside from that, FEEL expression represents a powerful tool that can be used in various cases, as described in detail in [56].

4.4.3 Multi-instance

A multi-instance is one of the constructs in Camunda 8 that is used to run an activity multiple times. When an activity is marked as a multi-instance using the Camunda Modeler, it basically works as a for-each loop in standard programming languages. The multi-instance can be any service and receive tasks, embedded sub-processes, or call activities.

On the execution level, the multi-instance activity consists of two parts. The first part is labeled multi-instance body and can be considered a container for the instances of the second part - the inner activity, as described in [57]. Once the activity is entered, the multi-instance body is activated, and one instance is created for each element of the *inputCollection*. The activity is left when each of the instances is completed.

The execution of multi-instance inner activities, which are independent of each other, can be done either sequentially or in parallel. As the name suggests, during a parallel run, multiple instances are executed concurrently. On the contrary, in the case of the sequential multi-instance activity, the instances are executed one at a time. When an instance is completed successfully, the next one can be created for the following element in *inputCollection*, as defined in [57].

The *inputCollection* must contain an expression that defines the collection to iterate over. To run without an incident (in Camunda Platform 8, an incident represents a problem that occurs during a process execution), the expression that is evaluated once the multi-instance body is activated must result in an *array* of any supported type. To access the current element of the *inputCollection* value within the instance, *inputElement* can be specified. Following that, the element is stored as a local variable of the inner activity instance under the determined name.

Similarly, the collection of the output is done. Multi-instance defines *outputCollection* and *outputElement* expressions in order to collect the output values. The *outputCollection* defines a name for the variable under which the output is stored once the multi-instance body is completed. The *outputElement* is used to store the result of each instance. It is created as a local variable of the instance. To collect the result of the multi-instance activity, this variable must be updated with the desired output value. When the instance is completed, the *outputElement* expression is evaluated, and the result is inserted into *outputCollection*.

In the presented solution, the sub-processes *Testimony creation* and *Proceedings participation* are marked as sequential multi-instances to be able to create multiple testimonies and involve multiple participants, respectively. Following the creation of *outputCollections testimonyList* and *participantContributionList*, the service tasks *Resolve testimonies* and *Resolve participant contributions* are used to transform the resulting collections into a form that can be inserted into a final decision. As the collections in both cases are *arrays* of Strings, the result of the transformation is a single formatted String with determined structure that can be directly used during the official decision generation.

4.5 Deployment

As described before, the implementation of the proof-of-concept BPM system in this case study is based on the Self-Managed version of Camunda Platform 8. More specifically, the entire environment consisting of the Zeebe engine, Identity, Operate, Tasklist, Optimize, Connectors, Elasticsearch, and Keycloak is run locally using Docker. In other words, the resulting BPM system is a locally running multi-container Docker application.

To define and run these applications, Camunda Platform 8 provides a Docker Compose file (a YAML file used to configure Docker images). In addition to that, the resulting configuration of the Zeebe Docker image can also be changed using environment variables, as described in [58]. To start the environment, the Docker Compose command must be issued.

Once the Docker images are running, the BPMN diagram representing the modeled process can be deployed to the local environment. The deployment of the BPMN executable model is done using the Desktop Modeler and the Zeebe gateway. Similarly, the instruction to start the process instance can be sent to the Zeebe engine after the process is successfully deployed.

4.6 Job workers

The executable BPMN model presented in the previous sections, together with the definitions of variables, expressions, Camunda Forms assigned to user tasks, and additional implementation and configuration details, can already be deployed to the Zeebe engine. However, since the model contains several script and service tasks, the execution of the process instance will never reach the end event. As mentioned above, the Zeebe brokers responsible for process execution do not include any business logic. Therefore, when script or service tasks are part of the process model, execution stops at the point where the task is entered. The Zeebe engine creates a job corresponding to the completion of the task and waits until the job is completed[2], as described in [60].

To complete a job, a job worker is used. In other words, a job worker is a service capable of performing a particular task represented by a job. Job workers request jobs of a certain type at regular intervals, which is called *polling*. The number of requested jobs, as well as the interval, is configurable in the Zeebe client. If one or more jobs of the requested type are available, the Zeebe engine will stream the activated job to the worker. When the job is received, the worker performs the business logic to complete it and sends back a *complete* or *fail* command depending on the possibility of completing the job successfully, as described in [61].

To scale up the processing, many workers can request the same job type. In this case, the Zeebe engine ensures that each job is sent to only one of the available job workers. Since job workers must repeatedly request jobs to find one to work on, both the engine and the workers perform a lot of unproductive work, which is expensive in terms of resource usage. To eliminate unnecessary work and ensure better utilization of resources, the Zeebe engine supports *long polling*. With *long polling*, a request is kept open while no jobs are available. When at least one job is created, the request is answered, and the job can be sent to job worker, as defined in [61].

The Zeebe engine also supports job queuing, which enables the decoupling of job creation and execution. It is always feasible to create a new job, regardless of the actual job worker unavailability. This is possible because Zeebe queues the jobs until workers request them. Additionally, this increases the resilience of the overall system. Since Camunda Platform 8 is highly available, job workers do not need to be. Zeebe can queue jobs during any job worker outages, so the progress will resume once the workers come back online.

4.6.1 Decision generating

Aside from the job workers used to perform the *Initialize creation* script task responsible for generating the unique ID of the process based on the name of the tax subject and the current timestamp, as well as the already described *Resolve testimonies* and *Resolve participant contri-*

[2]In the most recent version of Camunda Platform 8 – Camunda Platform 8.2 it is also possible to implement a script task using the FEEL expression instead of a job worker as specifies Fromme [59].

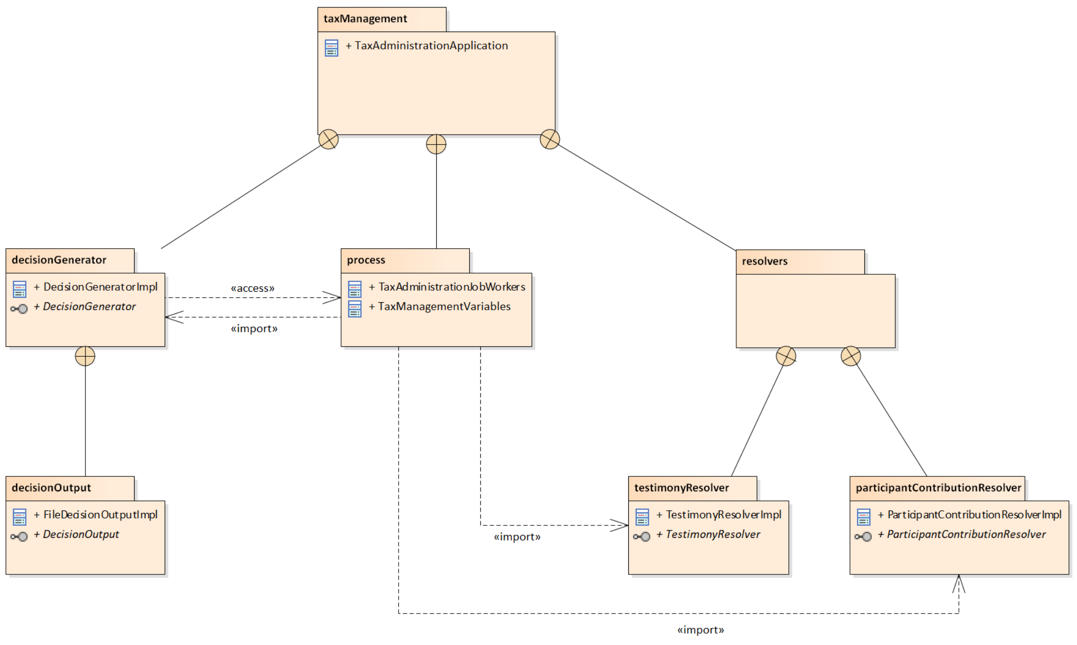


Figure 4.7 UML Package diagram of Zeebe client responsible for completing script and service tasks.

butions script tasks, the main job worker is in control of generating the official decision based on the information and values obtained during the execution of the process.

To implement the job workers required to complete the jobs created by script and service tasks, a Spring Boot application, whose structure can be seen in the Package diagram in Figure 4.7, representing the Zeebe client is created. To use Zeebe as part of the application, the `spring-zeebe-starter` dependency must be added. To configure the connection to the Zeebe broker, the `resources` folder must include an `application.properties` file that specifies it. The specification of the properties defining the connection, when using a locally running Zeebe engine, can be seen in Code listing 4.1. Firstly, the address of the gateway is defined. As the presented BPM system is based on the Self-Managed version of Camunda Platform 8, the process instance is running on *localhost*. Specifically, the Zeebe gateway is located on port `26500`.

After the connection to the broker is established, the application can be connected to Zeebe by using the `@EnableZeebeClient` annotation, which enables the injection of the Zeebe client. Next, the `@Deployment` annotation is defined. This annotation internally uses the Spring resource loader to deploy the process model. The resulting definition of the application's main class `TaxAdministrationApplication` is shown in Code listing 4.2.

Using this class definition, the Spring Boot application is now considered a Zeebe client. To implement job workers, the `TaxAdministrationJobWorkers` class is created. Job workers are the methods of this class labeled with a `@jobWorker` annotation. To specify the job type, the annotation takes a `type` attribute. To ensure that the job worker requests the demanded job, the value passed to this attribute matches the value defined in the executable BPMN model, specifically the attribute type in service task properties.

```
zeebe.client.broker.gateway-address=127.0.0.1:26500
zeebe.client.security.plaintext=true
```

Code listing 4.1 Properties specifiing connection to locally running Zeebe engine.

```java
@SpringBootApplication
@EnableZeebeClient
@Deployment(resources = "classpath:tax_management.bpmn")
public class TaxAdministrationApplication {

    public static void main(final String... args) {
        SpringApplication.run(TaxAdministrationApplication.class, args);
    }
}
```

Code listing 4.2 TaxAdministrationApplication class definition.

As some of the process variables may be necessary to complete the job, the variables can be passed to the Spring Boot application method representing the job worker. This can be done by using the `@Variable` annotation when defining the input variable of the method. If the name of the input variable corresponds to the process instance variable, its value is automatically fetched to the input variable. A second option to access the process variables from the job worker is much more useful if the number of necessary variables is higher. In this case, a separate class is specified that contains the required variables as standard Java variables. Each of the variables must have a defined getter and a specific setter that are used to map the class variables to the process instance variables and vice versa. Using this approach, the variables are loaded after labeling the input variable (its type corresponds to the defined custom variable class) with the *@VariablesAsType* annotation.

To return a value to the process instance, the return type of the job worker method is typically `Map<String, Object>`. The String value represents the name of the process variable, while the Object value is the returned object.

The definition of job worker that is responsible for generating the official decision based on the process instance variables and saving it to the file with a filename corresponding to the ID of the process can be seen in Code listing 4.3.

To generate the PDF document, the `DecisionGenerator` class that contains a public method `generate` is created. The method is responsible for the creation of a PDF file with a given name and the assembling of the official decision using the values and information gained during the process execution. The creation of the PDF is based on the methods of the *OpenPDF* library.

```java
@JobWorker(type = "generatingDecision")
public Map<String, Object> handleGeneratingDecisionJob(final ActivatedJob job,
     @VariablesAsType TaxManagementVariables variables) {
    String fileName = "results/TMD_" + variables.getDecisionId() + ".pdf";

    //generate pdf file
    decisionGenerator.generate(fileName, variables);
    log.info("Generated official decision, saved with filename: " + fileName);

    return Collections.singletonMap("decisionFileName", fileName);
}
```

Code listing 4.3 Job worker responsible for generating the official decision.

4.7 Testing

In order to ensure correct behavior and error-free execution, testing is considered an important part of software development. According to Celerity [62], finding errors and inaccuracies during the testing stage is 6 to 7 times less expensive than resolving problems in the production stage of a project. In other words, a production-ready system should be well-tested.

The main goal of the thesis is to find out how the presented methodology can be applied to digitize legal processes. The implementation of the proof-of-concept BPM system aims to be as thorough as possible, so that the resulting system can be close to real-world production systems. Because of this, the testing phase should be part of the proof-of-concept BPM system development process.

4.7.1 Process testing

To ensure the quality of the proof-of-concept BPM system, unit tests are used. As the implementation is partly based on the Spring Boot application, testing of the system is also done using the Spring Boot framework. To write tests that control the quality of the system depending on the Zeebe engine and the Zeebe client, the `spring-zeebe-test` dependency is added. This dependency enables the creation of test cases that focus on the correct behavior of the system represented by the Zeebe engine running the executable BPMN model together with the Zeebe client and the corresponding job workers.

The test classes are annotated using the `@SpringBootTest` and `@ZeebeSpringTest` annotations. As the first annotation is a primary annotation to create unit tests in Spring Boot applications, `@ZeebeSpringTest` enables the testing of executable BPMN process definition together with the glue code that logically belongs to the process definition in a wider sense, as described in [63]. The glue code tested together with the BPMN process definition is typically:

- Job workers code, usually connected to service or script tasks.
- FEEL expressions used in the process model for gateway decisions or input/output mappings
- Additional code (used typically for data mapping and delegating to the workflow engine)

As the class is annotated with `@ZeebeSpringTest`, the variables of types `ZeebeTestEngine` and `ZeebeClient` are automatically injected using the `@Autowired` annotation of Spring Boot. These variables are further used to control and eventually update the state of the engine and client. `ZeebeTestEngine` is an in-memory process engine that provides functions to help with writing the tests. `ZeebeClient` represents the client used to send the commands to the engine, such as starting the process instance.

The unit tests that are used to test the correct behavior of the system are based on the JUnit 5 testing framework. This framework is well-known and widely used to create unit tests on the JVM. The framework relies heavily on annotations, as written in [64].

First, the possibility to correctly deploy a process is tested. The testing is based on sending the deploy command via the client to the Zeebe engine. Next, the result of the command is verified using the `BpmnAssert` class and its method `assertThat()` that takes the resulting `DeploymentEvent`. The deployment test can be seen in Code listing 4.4.

After the correct deployment is tested, the verification of errorless execution of the process takes place. The verification is based on testing the process in chunks, as suggested in [63]. The whole process is divided into paths that are tested separately. One of the paths is usually the *happy path*, followed by testing of the detours. This ensures that the whole process is tested properly, meaning that every activity can be covered with a test that confirms its correct behavior and eliminates unexpected results.

```
@Test
public void testProcessDeployment() {
    // Create and send new deployment command
    DeploymentEvent deploymentEvent = zeebe.newDeployCommand()
            .addResourceFromClasspath("tax_management.bpmn")
            .send()
            .join();

    // Assert the correct deployment
    BpmnAssert.assertThat(deploymentEvent);
}
```

Code listing 4.4 Testing method used for verifying the errorless deployment of the process.

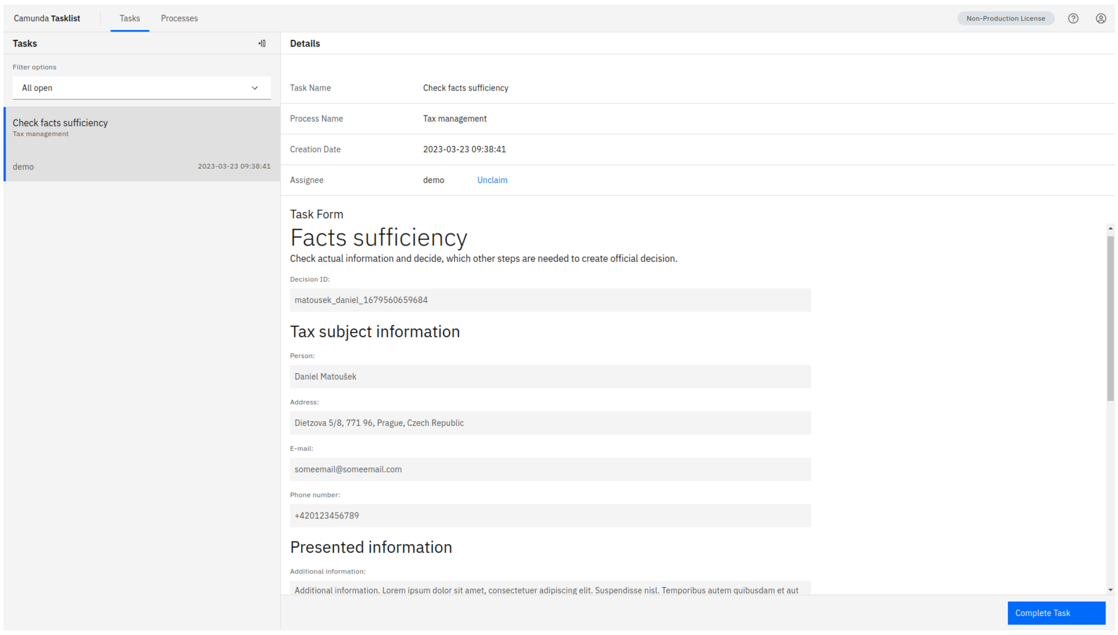
Testing process execution starts with the creation of the process instance, which is done using `newCreateInstanceCommand()` that initialize the process instance with a given `processId`. Once the instance is created, the process can be executed while the correct and expected behavior is tested.

As user tasks are completed by the user, their performance during testing must be done manually within the code. To simplify the completion of user tasks across testing methods, the `waitForUserTaskAndComplete()` method is defined. The method takes as a parameter the user task identifier, along with the `Map<String, Object>` representing the values that the user task should return. The execution of the method starts with the `waitForIdleState()` method called on `ZeebeTestEngine`, which ensures that the engine is able to perform all necessary actions before the user task is created. With that done, the jobs that represent incomplete user tasks can be listed. Once the correct job is obtained, the method verifies, using JUnit's `assertTrue` method, that the job corresponds to the determined user task with the given identifier. Next, the job can be completed using the `newCompleteCommand()` method, which takes the `key` of the user task job as a parameter. The return values are passed to the command using the `variables()` method.

As the process and the glue code are tested together, to prevent the generation of a PDF file every time the tests are executed, the `DecisionGenerator` is mocked. The mocking is based on creating an object that simulates the behavior of the real object without the side effects, which in this case is generating (that is tested separately) and storing the official decision file. To simulate the behavior of the `DecisionGenerator` class, the *Mockito* library and its built-in methods are used.

The final step of the test methods consists of verifying the correct process execution. To wait for process instance completion, the `waitForProcessInstanceCompleted()` method is used, taking as a parameter a reference to the current process instance. Next, assertions are executed to check whether the process instance has been completed successfully, whether there were any incidents during the process execution, and finally whether the process instance ended in the desired end event that corresponds to the tested path. This verification is done using the `hasPassedElement()` method, which takes as a parameter the identifier of the event, gateway, or activity that is expected to be passed, in this case the identifier of the end event.

The result of the testing is verified behavior of the system during the process execution, as well as tested possibility to deploy the process flawlessly. As the *happy path* and every detour are covered with tests and the correct behavior of the job workers and other glue code is checked, the system should be free from the vast majority of possible errors.



Check facts sufficiency
Tax management
demo 2023-03-23 09:38:41

Details

Task Name	Check facts sufficiency
Process Name	Tax management
Creation Date	2023-03-23 09:38:41
Assignee	demo Unclaim

Task Form

Facts sufficiency

Check actual information and decide, which other steps are needed to create official decision.

Decision ID: matousek_daniel_1679560659684

Tax subject information

Person: Daniel Matoušek

Address: Dietzova 5/8, 771 96, Prague, Czech Republic

E-mail: someemail@someemail.com

Phone number: +420123456789

Presented information

Additional information: Additional information. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Suspendisse nisl. Temporibus autem quibusdam et aut

Figure 4.8 Camunda Platform 8 Tasklist with form example.

4.8 Application prototype

The executable BPMN model, containing all the necessary implementation details, Camunda Platform 8 components running within a Docker and Spring Boot application, which is used to complete the service and script task jobs, create a prototype of the application. This prototype serves as a proof-of-concept BPM system that supports the tax administration process defined in the Czech code of law.

The system primarily helps with the orchestration of tasks and simplifies interactions between human actors who are responsible for completing them. As a result, the workload can be distributed among several users. Important information is stored during the process execution, making it available to the tax administrator when determining the tax amount and stating the arguments that lead to it. Additionally, some necessary steps are automated, which, combined with other factors, can accelerate the overall execution of the process.

Users of the presented system will interact primarily with the Tasklist component. Using the Tasklist web interface, users can fill out the information that is important for completing the tax administration's goal, i.e., determining the amount of tax to be paid by the tax subject and creating and publishing the official decision. During the execution of the process, the Tasklist generates the corresponding form based on the created Camunda Form whenever the user task needs to be completed. The user can claim the task, fill in the information, and submit it to the system. An example of the form in the Tasklist web interface is shown in Figure 4.8.

The interaction with the Tasklist begins with the insertion of the tax subject's personal information into a prepared form. Based on the submitted information and the current timestamp, a unique identifier for the process instance is generated by the system. Next, the user fills in additional information and presented facts obtained from the tax subject. After the validation of the facts and possible modification, the user is asked to read the available information and decide on its sufficiency or select required additional steps using checkboxes.

As mentioned in previous chapters, the tax administrator can opt for the creation of witness testimonies, expert reports, special records, or the participation of another person if necessary. These sub-processes are based on filling out information about the acts to prepared forms. Initially, the corresponding persons must be notified about the actions. Since the code of law demands notification in the form of registered mail with the executor's signature, the notification must be sent manually. Once the step is done, the person who deals with its completion submits the corresponding information to the system via prepared form. In the case of proceedings participation, the completion also consists of a compensation payment if the participation supervisor decides the claim. Additionally, tax negotiation can also supplement these steps in case the added information is still insufficient for the determination of the tax.

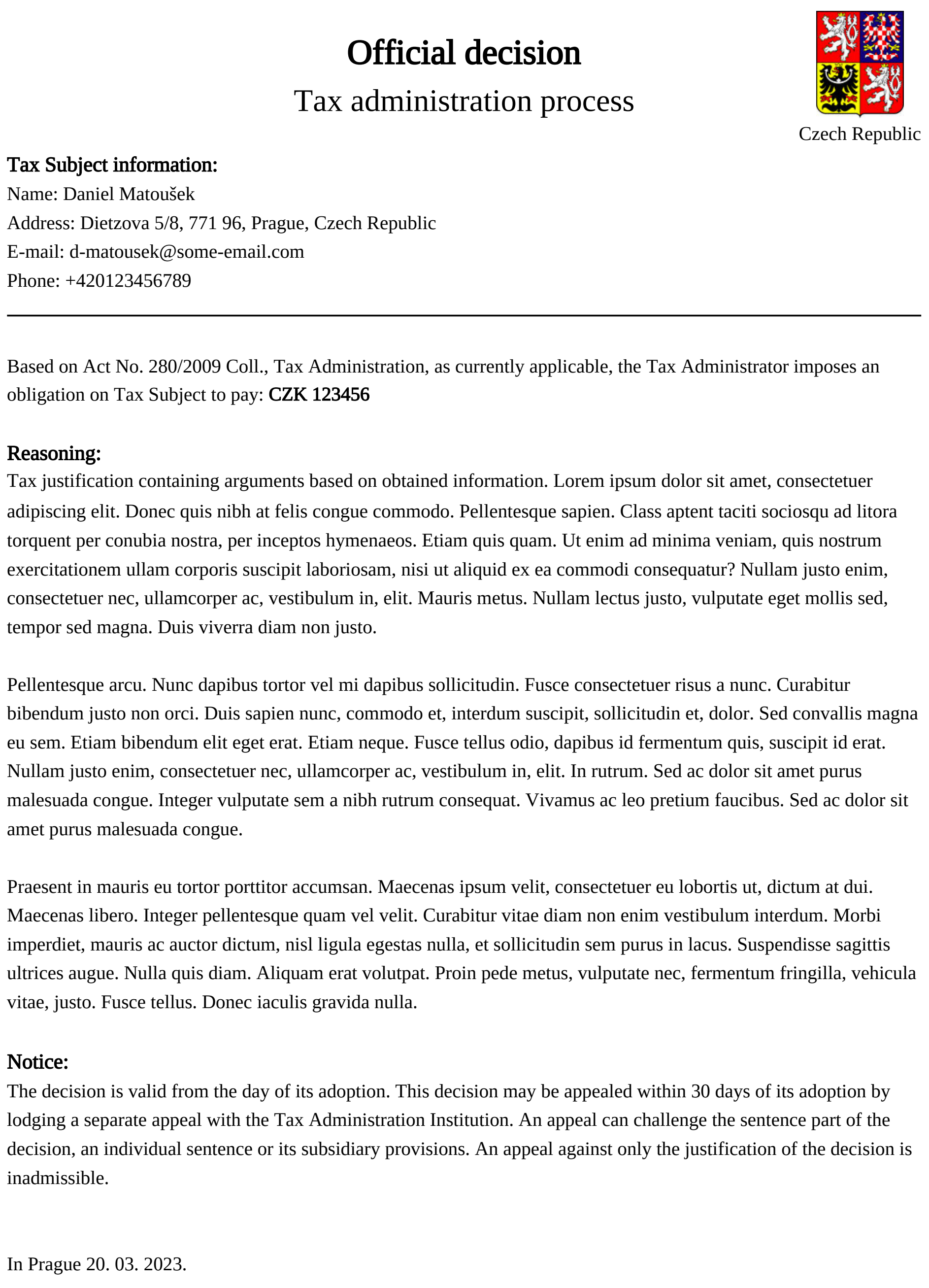
Once all the required information is obtained, the phase of official decision creation can begin. Firstly, the user checks if all the submitted information is correct. In the case of any content or typographical errors, the user can update the values before the final decision is created. To generate the decision, the tax administrator must submit the determined tax amount as well as its justification. The final assembly of the PDF representing the official decision, which is based on the values submitted in previous tasks, is automated. An example of generated decision can be seen in Figure 4.9. At the end of the process execution, the user is asked to publish the decision and get the proceedings cost paid by the tax subject.

In order for users to use the Tasklist component to complete tasks, they must first log in using the Identity component. Users log in with their credentials, typically their username or email address and password. To manage users in Camunda Platform 8 Self-Managed, the Keycloak web interface is used. Users can be assigned to groups based on their abilities and powers, and task completion can be directed to specific users through group assignments. For example, only users with the required education can create an expert opinion, so the creation of an expert report cannot be done by anyone who can log in to the Tasklist system, but only by users assigned to the *expert-report-creator* group. Alternatively, permissions to use Camunda Platform 8 components can be added to users, which can, for example, separate users at higher positions with access to the Optimize component from officials responsible for official decision creation, who primarily use Operate or Tasklist.

The Operate component can be used to monitor the current states of process instances as users gradually complete tasks. This enables users to view and control the execution of processes. Aside from viewing the current states of instances, the Operate web interface can also be used to view and modify process variables. The most important role of the Operate component is based on its ability to resolve incidents or cancel the entire process instance.

Figure 4.10 shows an example of overview of a currently running process instance in Operate. As can be seen, the process instance is currently waiting for the completion of two tasks, while simultaneously special records are being captured. In addition to the current state, the whole path of process execution is highlighted in different color in the overview. The instance history, which contains every activity and gateway passed so far during execution, is also part of the process detail, along with the overview of current variables and their values. As the process instance is still active, it is possible to update the variable values or add new variables. To thoroughly monitor the process execution, the Operate component offers the possibility to view details of completed instances and instances that ended up with an incident as well. This information can be subsequently used to resolve any problems and eliminate the possibility of another process instance failing.

While Operate focuses mainly on monitoring and controlling individual process instances, Optimize is used to create summary reports and dashboards based on information gained during numerous process executions for a defined period of time. This enables continuous process improvement by providing transparency and insight into designed workflows. The dashboards that can be built and customized directly using the Optimize web interface are useful mainly to users in higher positions and those responsible for the process performance, as they can provide important feedback about what is happening during the execution and what is slowing the process down. Information such as the number of process instances in a given period of time, the average duration of execution, the steps that are executed frequently, and the average time spent on tasks is available to users with access to Optimize.



Official decision

Tax administration process

Czech Republic

Tax Subject information:
Name: Daniel Matoušek
Address: Dietzova 5/8, 771 96, Prague, Czech Republic
E-mail: d-matousek@some-email.com
Phone: +420123456789

Based on Act No. 280/2009 Coll., Tax Administration, as currently applicable, the Tax Administrator imposes an obligation on Tax Subject to pay: **CZK 123456**

Reasoning:
Tax justification containing arguments based on obtained information. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec quis nibh at felis congue commodo. Pellentesque sapien. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Etiam quis quam. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Nullam justo enim, consectetuer nec, ullamcorper ac, vestibulum in, elit. Mauris metus. Nullam lectus justo, vulputate eget mollis sed, tempor sed magna. Duis viverra diam non justo.

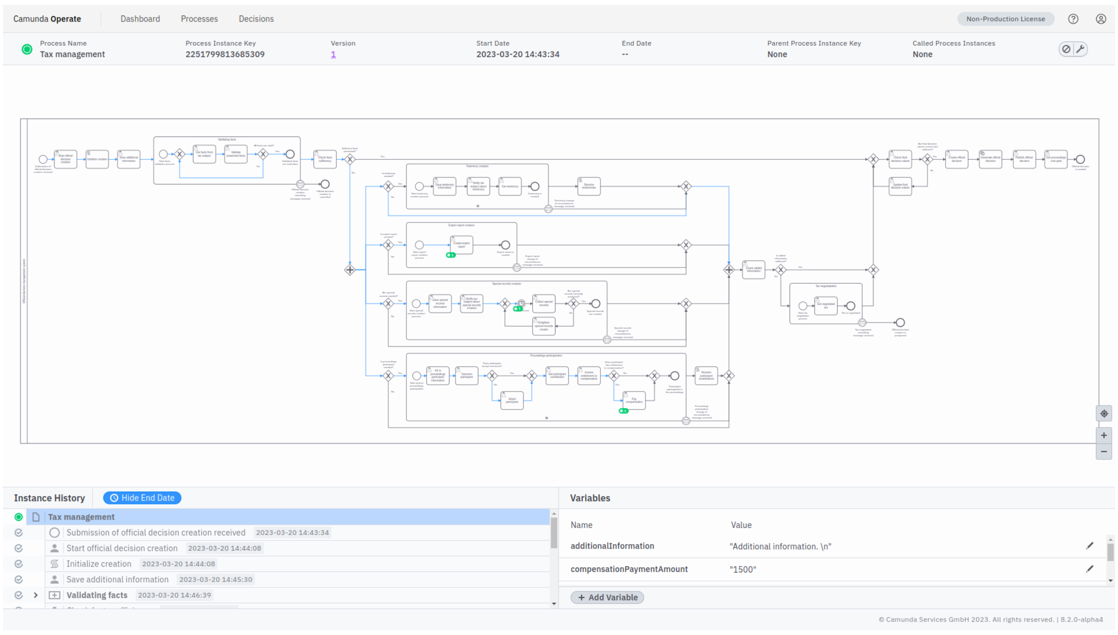
Pellentesque arcu. Nunc dapibus tortor vel mi dapibus sollicitudin. Fusce consectetuer risus a nunc. Curabitur bibendum justo non orci. Duis sapien nunc, commodo et, interdum suscipit, sollicitudin et, dolor. Sed convallis magna eu sem. Etiam bibendum elit eget erat. Etiam neque. Fusce tellus odio, dapibus id fermentum quis, suscipit id erat. Nullam justo enim, consectetuer nec, ullamcorper ac, vestibulum in, elit. In rutrum. Sed ac dolor sit amet purus malesuada congue. Integer vulputate sem a nibh rutrum consequat. Vivamus ac leo pretium faucibus. Sed ac dolor sit amet purus malesuada congue.

Praesent in mauris eu tortor porttitor accumsan. Maecenas ipsum velit, consectetuer eu lobortis ut, dictum at dui. Maecenas libero. Integer pellentesque quam vel velit. Curabitur vitae diam non enim vestibulum interdum. Morbi imperdiet, mauris ac auctor dictum, nisl ligula egestas nulla, et sollicitudin sem purus in lacus. Suspendisse sagittis ultrices augue. Nulla quis diam. Aliquam erat volutpat. Proin pede metus, vulputate nec, fermentum fringilla, vehicula vitae, justo. Fusce tellus. Donec iaculis gravida nulla.

Notice:
The decision is valid from the day of its adoption. This decision may be appealed within 30 days of its adoption by lodging a separate appeal with the Tax Administration Institution. An appeal can challenge the sentence part of the decision, an individual sentence or its subsidiary provisions. An appeal against only the justification of the decision is inadmissible.

In Prague 20. 03. 2023.

Figure 4.9 Example of generated official decision.



Process Name	Process Instance Key	Version	Start Date	End Date	Parent Process Instance Key	Called Process Instances
Tax management	2251799813685309	1	2023-03-20 14:43:34	--	None	None

Instance History

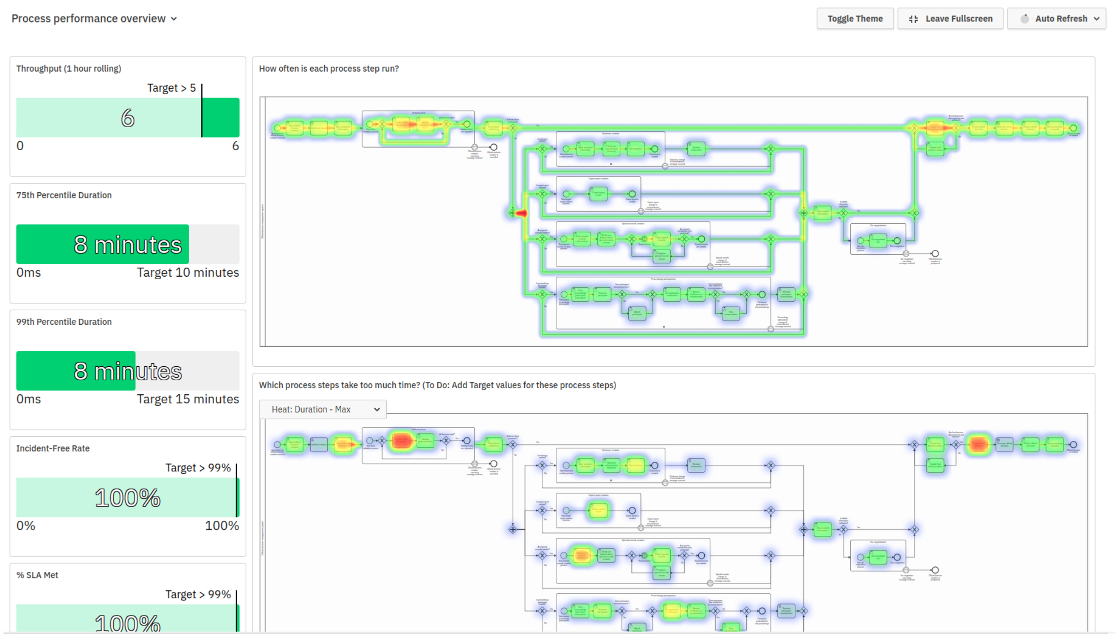
Tax management	
Submission of official decision creation received	2023-03-20 14:43:34
Start official decision creation	2023-03-20 14:44:08
Initialize creation	2023-03-20 14:44:08
Save additional information	2023-03-20 14:45:30
Validating facts	2023-03-20 14:46:39

Variables

Name	Value
additionalInformation	"Additional information. \n"
compensationPaymentAmount	"1500"

© Camunda Services GmbH 2023. All rights reserved. | 8.2.0-alpha4

Figure 4.10 Camunda Platform 8 Operate with example of process instances.



Throughput (1 hour rolling)
Target > 5
6
0 6

75th Percentile Duration
8 minutes
0ms Target 10 minutes

99th Percentile Duration
8 minutes
0ms Target 15 minutes

Incident-Free Rate
Target > 99%
100%
0% 100%

% SLA Met
Target > 99%
100%

How often is each process step run?

Which process steps take too much time? (To Do: Add Target values for these process steps)

Figure 4.11 Camunda Platform 8 Optimize dashboard example.

An example of the generated dashboard can be seen in Figure 4.11. As shown, the dashboard includes a numerical summary of throughput and duration, as well as the incident-free rate. Heatmaps can also play an important role in showing how often each step is executed and how long it takes to complete each step. By using the reports and dashboards, bottlenecks in the process can be detected, and necessary changes to the process model or personnel reassignments can be made.

4.9 Summary

The implementation has resulted in a proof-of-concept BPM system based on Camunda Platform 8, which supports the tax administration process. The overall system consists of several components with different functions and purposes. The main component is the Zeebe engine, which is responsible for process execution and communication with the Zeebe client that contains the job workers. Tasklist is important component for users, as it acts as a mediator between the process engine and users by allowing them to complete user tasks through designed forms. In addition to that, Operate and Optimize components enable the possibility to monitor instances and improve the process.

The main benefit of the system corresponds to its ability to manage the workflow, help with orchestration, and simplify the interaction between users. In addition, the system automates some necessary steps to accelerate the execution of the entire tax administration process.

The testing phase of the development process involves verifying the correct behavior and incident-free execution of the created system. This is achieved through written unit tests that cover 100% of the activities, including additional code such as decision creation, to ensure system quality and eliminate errors and bugs that may occur when users interact with the system. These tests are written with the help of the `@SpringBootTest` and `@ZeebeSpringTest` annotations, which set up the application context for testing and enable in-memory testing of the process running in the Zeebe engine, respectively.

Chapter 5

Case study evaluation

The goal of the thesis, which is based on the creation of the proof-of-concept BPM system using technologies researched at the Czech Technical University, was accomplished. The system supporting the tax administration process defined in the Czech code of law was implemented using the Camunda Platform 8. The implementation was grounded on the as-is analysis of the current state that used the DEMO methodology to reveal the essence of the system, as well as the to-be analysis using the BPMN to design a to-be state representing the way of the process execution after the deployment of the BPM system.

This chapter focuses on the evaluation of the system development process. In the following sections, key findings and mainly the advantages and disadvantages of the selected development approach will be described. Aside from that, the limitations of the presented system and the future development possibilities and improvement options of the proposed BPM system, which in most cases are currently not possible to implement in order to comply with the legislation, will be presented.

5.1 Key findings

As the goal of the thesis was achieved successfully, the presented approach to develop a BPM system that supports the process defined in the code of law can be considered successful. The involvement of the BPMN and DEMO methodology can, to a certain degree, simplify the overall process of system creation.

5.1.1 Advantages

To conclude the selected approach to create the system, firstly, the main benefits of the methodology are described. The following aspects that correspond to the usage of DEMO, BPMN, and Camunda Platform during the analysis, design, and implementation processes were found to be the most beneficial:

- **DEMO's capability to analyse the process thoroughly**

  One of the most significant advantages of the DEMO methodology is its ability to reveal the essence of the process. After the analysis using the DEMO methodology and its OER analysis is completed, a detailed overview of actors, together with transactions containing information about each individual step of the process, as well as about the final products, is created. Additionally, missing definitions of steps are clearly visible. Once the DEMO analysis is done, the formal foundations for further BPMN models are created. As the final result is very comprehensive, the BPMN activities, important data, and process orchestration are basically revealed, and the person creating the BPMN model has a clear picture of the overall process and understands its functioning.

- **Initial process analysis from a structural perspective**

  Typically, BPMN processes are modeled in a chronological order of activities, from the starting point to the end point. However, this can become problematic when modeling procedural law processes, as the definitions of the activities are typically not ordered consecutively within the code of law. In contrast, the DEMO methodology, and especially the OER analysis approach to reveal the essence of the process, is based on firstly identifying all the transactions. These transactions can be easily reordered later on, since DEMO analyses the process from a structural perspective.

- **BPMN's simplicity and understandability**

  Since BPMN is a well-known, widely used, and relatively easy to learn notation, the models created using it are generally understandable. This can simplify the communication with people who will be using the system, as well as within the development team. Even those who have never encountered notation before should be able to imagine the process execution when presented with a BPMN model.

- **Possibility to create the BPM system using the executable BPMN model**

  Using the Camunda Platform or any other engine that can run executable BPMN model simplifies the development process, as it eliminates the need for time-consuming code writing. To create a proof-of-concept BPM system using this approach, the implementation details and necessary configurations must be added to the existing analytical model, so that the system can be successfully deployed to the engine. In addition, client and job worker implementations must be created. However, the overall implementation phase of the development can be expedited in comparison to creating the system from scratch.

5.1.2 Disadvantages

Even though the selected approach led to the creation of the proof-of-concept BPM system, some disadvantages of the methodology were found. To objectively evaluate the development process, the main drawbacks, together with a short explanation, are presented in the following list:

- **Complexity of DEMO methodology**

  In general, the OER steps, which help to reveal the essence of the process and create the DEMO models, are defined in detail. If the steps of the OER analysis are followed, the result also complies with the techniques used to achieve the essence and simplicity in DEMO. The creation of the DEMO models follows strict rules in order to ensure their correctness. However, strictness during the process of essence revealing and model creation can be limiting in some cases. Although it is necessary to create correct models, the rules do not allow for any sort of freedom during the analysis. For instance, if the analyst wants to omit some information from the models as it is not needed in further development, from the perspective of DEMO methodology, the result is incomplete and it cannot be considered correct.

- **Inconsistency and ambiguity of BPMN models**

  As BPMN is an easy-to-use notation with many distinct use cases corresponding to the different levels of detail contained in the model, the notation must offer flexibility. However, in some cases, this flexibility can result in inconsistencies or ambiguities in the process models. It is possible to create many different models that are equally correct solutions to a modeled process. For example, the modeling of an action that needs to be repeated multiple times, such as obtaining testimonies from multiple witnesses, can be considered. In this case, it is possible to model the situation using the multi-instance construct, loop construct, or standard sequence flow and gateways as shown in a similar example in [65]. This means that there are three possible solutions for a single part of the process. This is one of the reasons why the usual process of creating a single solution often results in the repeated, time-consuming designing of several versions of BPMN model that are identical from the process execution perspective.

5.1.3 Recommendations

Despite the fact that the presented methodology has some disadvantages, it can be considered effective overall. The benefits of each step in the development process outweigh the drawbacks, particularly when creating a BPM system to support legal processes.

Even though the DEMO methodology is complex and strict, a detailed overview of transactions, actors, and data that occur during the process execution is essential for further development. It creates a solid foundation for subsequent creation of the BPMN models that is less demanding. The same is true for the second step, designing the to-be state of the process. The analytical BPMN model resulting from the to-be analysis is a useful starting point for creating the executable BPMN model. Therefore, it is worth spending the necessary time to complete the steps.

Despite thorough analysis, ensuring that the BPM system is fully compliant with relevant legal requirements and regulations is critical when dealing with legislation. Legal experts should check if the system complies with regulations and if the system is not based on a wrong interpretation of the law before using it in production. Even better would be close cooperation with legal experts during the analysis, design, and implementation phases of the development process to ensure that the system complies with all relevant laws, requirements, and guidelines from the beginning.

5.2 System limitations

Although the proof-of-concept BPM system aims to be as close to production as possible, the result requires additional implementation to be able to deploy it to production. To function effectively, the production-ready BPM system will need to integrate with various other systems, such as tax calculation software, financial management systems, and possibly reporting tools.

Aside from that, the resulting proof-of-concept BPM system was run only locally using the Docker Compose file, which is not recommended for production usage as it is not optimized. The production version of the system requires the usage of Kubernetes or available Docker images when using the Self-Managed Camunda Platform 8, or eventually consider the shift to a SaaS solution. To use the system in production in the presented form, both approaches require paying the monthly fee that depends on various variables, such as the number of users or process instances, as described in [66].

5.3 Future development

As already mentioned, the digitization of the tax administration process is limited in order to comply with the definition in the Czech code of law. Currently, the code of law contains prescribed procedures and aspects that create barriers to thorough digitization of the processes. Since the system must comply with the legislation, it must follow the defined procedures and requirements. In order to allow further digitization, the corresponding changes must be enacted simultaneously with the development of the system.

One of the aspects that creates a barrier for thorough digitization is based on the necessity to notify the parties to the proceedings using registered mail containing manually signed documents. In the case of the tax administration process, this means that every time the tax subject, proceedings participant, or any other party to the proceedings needs to be notified about anything, the tax administrator must create a document, print, sign, and send it manually as registered mail. The only possible solution to comply with current legislation and expedite the process execution is based on the creation of a separate department that would focus strictly on the need to create and send notifications by registered mail. That would enable the tax administrator to work on the tasks necessary for the tax determination and decision creation instead of distractions corresponding with the notifications.

In case the legislation enables notifying the party to the proceedings using email communication or alternatively eGovernment or other currently available technologies, the overall process execution can be expedited.

As email communication can be used as a tool to interact with parties, the system could automate the notification tasks. Because the Camunda Platform 8 contains the possibility to send email messages using the built-in Connector, the overall changes to the implementation would be undemanding. These changes would consist of the configuration of the SendGrid Email Connector that is responsible for email communication directly from the process. The configuration consists of the definition of the sender and recipient properties, as well as the creation of dynamic templates that specify the structure and form of the email.

Another possible option to automate the notification of parties is based on the use of the technology called *Datová schránka*. According to the Ministry of the Interior of the Czech Republic [67], the *Datová schránka* is a state-guaranteed communication tool that replaces classic registered mail and is mainly used for interactions with public authorities. To integrate the possibility to use this technology, the definition of a custom Connector that enables the engine to send messages using this technology directly from the process must be created. However, currently, the technology is not used enough among people to fully replace the need to send registered mail, since approximately only 5% of natural persons in the Czech Republic use the technology to communicate with public authorities or in some other way, as shown by data from the Ministry of the Interior of the Czech Republic [68].

Similar to the notification digitization situation are the payment automation circumstances. The payments are part of the tax administration process as the costs of proceedings need to be paid, or eventually if the proceedings participation is necessary and the participant has the entitlement to compensation. In this case, part of the participation sub-process consists of payment of the determined amount of compensation to the participant. These payments are currently done manually, as required by the legislation. To digitize the process, the automation of payments using the integration of some payment technology should be considered. Similarly, as in the case of notifications, task completion would be automated, and the tax administrator could focus on subsequent tasks. In view of that, the process execution could be expedited notably.

To achieve more advanced digitization, a system must be created that allows tax subjects, proceedings participants, and other relevant parties to submit information without interacting with the tax administrator. This can be accomplished by leveraging the existing eGovernment identity system to grant access to the system, which would enable users to fill in the required information directly. The tax administrator's participation would be limited to validating the submitted information and creating the official decision, as well as completing additional tasks that require his involvement.

However, transitioning to an almost fully automated system requires careful planning, attention to detail, and rigorous testing to ensure that it can handle a higher volume of users. The system must be designed in a way that people can understand how it works and how to use it effectively. This is necessary to avoid creating difficulties when interacting with a malfunctioning system and to ensure that the system can truly help and expedite the process execution.

Conclusion

The thesis has successfully achieved its goal of creating proof-of-concept BPM system to support the tax administration process defined in the Czech code of law. To accomplish this, a specific approach was taken that utilized the DEMO methodology to analyse the current state of the process, BPMN to design a future shape of the process after digitization, and Camunda Platform to implement the BPM system supporting the tax administration process based on the solution designed in the previous step.

The thesis is divided into five chapters. Firstly, an overview of law modeling and BPM systems was provided and current state-of-the-art BPMN, DEMO and Camunda Platform was reviewed. Next, the tax administration as-is state was analysed using DEMO methodology and its OER analysis. Utilizing this knowledge, the to-be state of the process after digitization captured by the analytical BPMN model was created. The following chapter describes the implementation of the BPM system based on the Camunda Platform 8. This chapter contains an overview of the hosting options and overall architecture of Camunda Platform 8 and its components, as well as a description of the executable BPMN model, the creation of Camunda Forms, and definitions of the implementation details. Additionally, the chapter presents the realization of the Zeebe client and job workers and shows possible way of the BPM system testing. Finally, the last chapter consists of a summary of the selected development approach, together with the advantages and disadvantages that the involvement of DEMO and BPMN brings. The chapter also discusses future development possibilities that could further enhance the effectiveness of the system.

The result of the development process is a proof-of-concept BPM system designed to support the tax administration process defined in the Czech code of law. The system is based on the Self-Managed version of Camunda Platform 8 running in Docker. The resulting system consists of deployed executable BPMN model, along with a Spring Boot application that represents the Zeebe client used to define the job workers. Users can interact with the system using the Tasklist component, which allows them to fill in important information related to the tax administration process using the forms generated on the basis of defined Camunda Forms. Additionally, the system contains Operate and Optimize, which enable users to monitor and control process execution and create summarizing reports and dashboards based on information gained during process executions, respectively. Authorization to the system is done and configured using Identity and Keycloak.

The main benefit of the system is based on the advanced orchestration of tasks and simplification of interactions between human actors, which allows redistribution of workloads. The system enables the storage of important information during the process execution to be available to tax administrator when the tax is determined and the reasoning of the amount is created. Aside from that, some of the necessary steps that do not require human intervention, such as the assembly of the final document representing the official decision of the tax administration process, are automated. The automation, together with the orchestration of tasks and redistribution of workloads, allows for further expedition of the overall execution of the tax administration process.

The thorough digitization of legal processes is currently limited by the prescribed procedures and requirements defined in the code of law. To comply with the legislation, the system must follow the regulations, which makes in-depth digitization impossible. In the case of the tax administration process, one of the aspects that create a barrier to thorough digitization is the demand to notify parties to the proceedings via registered mail containing manually signed documents. The notification could be easily done using email communication or other currently available technology, but its usage requires a change in legislation. The same can be said about other possible automation areas of the tax administration process.

In general, digitization has the potential to expedite process execution. However, since the law was not originally intended to be digitized, a further and more thorough digitization of legislative processes is contingent on corresponding changes to the legislation that must be enacted simultaneously with system development.

Bibliography

1. GABRYELCZYK, R.; BIERNIKOWICZ, A. *Motivations for BPM Adoption: Initial Taxonomy based on Online Success Stories.* Federated Conference on Computer Science and Information Systems (FedCSIS), Leipzig, Germany, 2019. Available from DOI: `10.15439/2019F229`.
2. ZORZANELLI COSTA, M.; GUIZZARDI, G.; ALMEIDA, J. P. A. On Capturing Legal Knowledge in Ontology and Process Models Combined. In: FRANCESCONI, E.; BORGES, G.; SORGE, Ch. (eds.). *Legal Knowledge and Information Systems.* IOS Press, 2022. ISBN 978-1-64368-365-2.
3. CIAGHI, A.; VILLAFIORITA, A. Improving Public Administrations via Law Modeling and BPR. In: POPESCU-ZELETIN, R.; RAI, I. A.; JONAS, K.; VILLAFIORITA, A. (eds.). *E-Infrastructure and E-Services for Developing Countries.* Springer, Berlin, Heidelberg, 2011. ISBN 978-3-642-23828-4.
4. *Zpráva o digitalizaci veřejné správy v České republice.* Nejvyšší kontrolní úřad. [online]. [Accessed: 2023-4-9]. `https://www.nku.cz/cz/publikace-a-dokumenty/ostatni-publikace/zprava-o-digitalizaci-verejne-spravy-v-ceske-republic-id10937/`.
5. *Registr vozidel se dočasně vrací k původnímu systému. Za nový ministerstvo zatím nezaplatí.* ČT24. [online]. [Accessed: 2023-4-9]. `https://ct24.ceskatelevize.cz/domaci/1159050-registr-vozidel-se-docasne-vraci-k-puvodnimu-systemu-za-novy-ministerstvo-zatim`.
6. *Statistický úřad chce prošetřit, proč nefungoval systém pro sčítání lidu. Audit ale zatím nezadal.* iROZHLAS. [online]. [Accessed: 2023-4-9]. `https://www.irozhlas.cz/zpravy-domov/scitani-lidu-2021-audit-vypadek-systemu_2104031831_zuj`.
7. *Rada vlády pro informační společnost: Program Digitální Česko.* Ministerstvo vnitra České republiky. [online]. [Accessed: 2023-4-9]. `https://www.mvcr.cz/webpm/clanek/rada-vlady-pro-informacni-spolecnost.aspx?q=Y2hudW09Ng%3D%3D`.
8. CIAGHI, A.; WELDEMARIAM, K.; VILLAFIORITA, A.; KESSLER, F. *Law Modeling with Ontological Support and BPMN: a Case Study.* IARIA, 2011. ISBN 978-1-61208-122-9.
9. VAN DER AALST, W. M. P. *Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management.* Springer, Berlin, Heidelberg, 2004. ISBN 978-3-540-27755-2.
10. WESKE, M. *Business Process Management: Concepts, Languages, Architectures.* Springer, Berlin, Heidelberg, 2012. ISBN 978-3-642-28616-2.
11. *Business Process Model and Notation (BPMN).* OMG. [online]. [Accessed: 2023-01-22]. `https://www.omg.org/spec/BPMN/2.0/PDF`.

12. MALEKAN, H. S.; SHAFAHI, M.; AYAT, N.; AFSARMANESH, H. *Enhancing Robust Execution of BPMN Process Diagrams: A Practical Approach.* Springer, Cham, 2018. ISBN 978-3-319-99127-6.
13. ALLWEYER, T. *Bpmn 2.0: Introduction to the Standard for Business Process Modeling.* BoD – Books on Demand, 2016. ISBN 9783837093315.
14. WALSER, K.; SCHAFFROTH, M. *BPM and BPMN as Integrating Concepts in eGovernment -: The Swiss eGovernment BPM Ecosystem.* Springer, Berlin, Heidelberg, 2011. ISBN 978-3-642-23135-3.
15. SILVER, B. *BPMN method and style: with BPMN implementer's guide.* Cody-Cassidy Press, 2011. ISBN 978-0982368114.
16. AAGESEN, G.; KROGSTIE, J. *BPMN 2.0 for Modeling Business Processes.* Springer, Berlin, Heidelberg, 2015. ISBN 978-3-642-45100-3.
17. DIETZ, J. L. G.; MULDER, H. B. F. *Enterprise Ontology: A Human-Centric Approach to Understanding the Essence of Organisation.* Springer, Cham, 2020. ISBN 978-3-030-38853-9.
18. *DEMO.* Enterprise Engineering Institute. [online]. [Accessed: 2022-12-15]. `https://ee-institute.org/demo/`.
19. DIETZ, J. L. G.; MULDER, J. B. F. *The Evolution of DEMO.* 2020.
20. VAN NUFFEL, D.; MULDER, H.; VAN KERVEL, S. *Advances in Enterprise Engineering III: Enhancing the Formal Foundations of BPMN by Enterprise Ontology.* Springer, Berlin, Heidelberg, 2009. ISBN 978-3-642-01915-9.
21. PERINFORMA, A. P. C. *The essence of organisation: An introduction to enterprise engineering.* Sapio Enterprise Engineering, 2017. ISBN 978-90-815449-4-8.
22. *What is Camunda Platform 8?* Camunda Platform 8 Docs. [online]. [Accessed: 2023-2-10]. `https://docs.camunda.io/docs/components/concepts/what-is-camunda-platform-8/`.
23. FOWLER, M.; LEWIS, J. *Microservices: a definition of this new architectural term.* 2014. [online]. [Accessed: 2023-2-10]. `https://martinfowler.com/articles/microservices.html`.
24. DRAGONI, N.; GIALLORENZO, S.; LLUCH LAFUENTE, A.; MAZZARA, M.; MONTESI, F.; MUSTAFIN, R.; SAFINA, L. Microservices: Yesterday, Today, and Tomorrow. In: MAZZARA, M.; MEYER, B. (eds.). *Present and Ulterior Software Engineering.* Springer, Cham, 2017. ISBN 978-3-319-67425-4.
25. *Camunda Platform 8: Software for Process Orchestration.* Camunda. [online]. [Accessed: 2023-2-10]. `https://camunda.com/platform/`.
26. FREUND, J. *Camunda Platform 8: A 10-Year Journey.* 2022. [online]. [Accessed: 2023-2-10]. `https://camunda.com/blog/2022/04/camunda-platform-8-a-10-year-journey/`.
27. VAN DER AALST, W. M. P. *Process Mining: Data Science in Action.* Springer, Berlin, Heidelberg, 2016. ISBN 978-3-662-49851-4.
28. SILVER, B. *BPMN, BPMS: Executable BPMN 2.0.* 2011. [online]. [Accessed: 2023-4-10]. `https://methodandstyle.com/executable-bpmn-2-0/`.
29. *Zákon č. 280/2009 Sb., zákon daňový řád, v platném znění.* [online]. 2009. [Accessed: 2023-01-22]. `https://www.zakonyprolidi.cz/cs/2009-280`.
30. TAVALSTO, A. *Introducing a New Modeler for Camunda Platform 8.* 2022. [online]. [Accessed: 2023-2-12]. `https://camunda.com/blog/2022/03/introducing-a-new-modeler-for-camunda-cloud/`.
31. *Model a BPMN 2.0 diagram.* Camunda Platform 8 Docs. [online]. [Accessed: 2023-2-12]. `https://docs.camunda.io/docs/components/modeler/bpmn/`.

32. *Modeling beyond the happy path*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-2-14]. `https://docs.camunda.io/docs/components/best-practices/modeling/modeling-beyond-the-happy-path/`.
33. *Get started with Camunda Platform 8: End-to-end process orchestration to transform your organization*. Camunda. [online]. [Accessed: 2023-2-18]. `https://camunda.com/get-started/`.
34. *Camunda Platform 8 Self-Managed*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-2-18]. `https://docs.camunda.io/docs/self-managed/about-self-managed/`.
35. LEVY, D. *Camunda Platform 8 – Orchestrate All the Things*. 2022. [online]. [Accessed: 2023-2-18]. `https://camunda.com/blog/2022/04/camunda-platform-8-orchestrate-all-the-things/`.
36. *Camunda Platform 8 installation overview*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-2-18]. `https://docs.camunda.io/docs/self-managed/platform-deployment/overview/`.
37. *Overview Components*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-4]. `https://docs.camunda.io/docs/components/`.
38. *Zeebe*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-4]. `https://docs.camunda.io/docs/components/zeebe/zeebe-overview/`.
39. *Architecture*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-4]. `https://docs.camunda.io/docs/components/zeebe/technical-concepts/architecture/`.
40. *What is Elasticsearch?* Elastic. [online]. [Accessed: 2023-3-4]. `https://www.elastic.co/what-is/elasticsearch`.
41. *What is Kibana?* Elastic. [online]. [Accessed: 2023-3-4]. `https://www.elastic.co/what-is/kibana`.
42. *Community Clients Overview*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-4]. `https://docs.camunda.io/docs/apis-clients/community-clients/`.
43. *What is Identity?* Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-5]. `https://docs.camunda.io/docs/self-managed/identity/what-is-identity/`.
44. *Server Administration Guide*. keycloak.org. [online]. [Accessed: 2023-3-5]. `https://www.keycloak.org/docs/16.1/server_admin/#using-the-admin-console`.
45. *Operate: Introduction*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-5]. `https://docs.camunda.io/docs/next/components/operate/operate-introduction/`.
46. *Tasklist: Introduction*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-5]. `https://docs.camunda.io/docs/components/tasklist/introduction-to-tasklist/`.
47. *What is Optimize?* Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-5]. `https://docs.camunda.io/optimize/components/what-is-optimize/`.
48. *Connectors: Introduction*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-9]. `https://docs.camunda.io/docs/components/connectors/introduction-to-connectors/`.
49. *Out-of-the-box Connectors: Overview*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-9]. `https://docs.camunda.io/docs/components/connectors/out-of-the-box-connectors/available-connectors-overview/`.
50. *Logical Data Model - UML Notation*. Sparx Systems. [online]. [Accessed: 2023-4-11]. `https://sparxsystems.com/resources/gallery/diagrams/software/sw-logical_data_model-uml_notation.html`.
51. *What are Camunda Forms?* Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-11]. `https://docs.camunda.io/docs/next/components/modeler/forms/camunda-forms-reference/`.

52. *Form Elements*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-11]. https://docs.camunda.io/docs/next/components/modeler/forms/form-element-library/forms-element-library/.
53. *Data binding*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-11]. https://docs.camunda.io/docs/next/components/modeler/forms/configuration/forms-config-data-binding/.
54. *Variables*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-12]. https://docs.camunda.io/docs/components/concepts/variables/.
55. *Expressions*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-12]. https://docs.camunda.io/docs/components/concepts/expressions/.
56. *FEEL Expressions Introduction*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-12]. https://docs.camunda.io/docs/components/modeler/feel/language-guide/feel-expressions-introduction/.
57. *Multi-instance*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-12]. https://docs.camunda.io/docs/next/components/modeler/bpmn/multi-instance/.
58. *Docker*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-9]. https://docs.camunda.io/docs/next/self-managed/platform-deployment/docker/.
59. FROMME, P. *Camunda Desktop Modeler 5.7 released*. 2023. [online]. [Accessed: 2023-3-13]. https://camunda.com/blog/2023/01/camunda-desktop-modeler-5-7/.
60. *Service tasks*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-13]. https://docs.camunda.io/docs/components/modeler/bpmn/service-tasks/.
61. *Job workers*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-13]. https://docs.camunda.io/docs/components/concepts/job-workers/.
62. *The True Cost of a Software Bug: Part One*. Celerity. [online]. [Accessed: 2023-3-18]. https://www.celerity.com/insights/the-true-cost-of-a-software-bug#.
63. *Testing process definitions: Writing process tests in Java*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-18]. https://docs.camunda.io/docs/next/components/best-practices/development/testing-process-definitions/.
64. *JUnit 5*. JUnit Team. [online]. [Accessed: 2023-3-18]. https://junit.org/junit5/.
65. *Modeling with situation patterns*. Camunda Platform 8 Docs. [online]. [Accessed: 2023-3-13]. https://docs.camunda.io/docs/components/best-practices/modeling/modeling-with-situation-patterns/.
66. *Camunda Platform 8 Pricing*. Camunda. [online]. [Accessed: 2023-4-8]. https://camunda.com/pricing/.
67. *Datové schránky*. Ministerstvo vnitra České republiky. [online]. [Accessed: 2023-3-21]. https://www.mvcr.cz/clanek/datove-schranky-datove-schranky.aspx.
68. *Zájem o datové schránky stále roste. Datovku si zřídilo už přes půl milionu občanů*. Ministerstvo vnitra České republiky. [online]. [Accessed: 2023-3-21]. https://www.mvcr.cz/clanek/zajem-o-datove-schranky-stale-roste-datovku-si-zridilo-uz-pres-pul-milionu-obcanu.aspx.

Contents of attachment

```
readme.txt ........ the file with attachments contents description
src
  thesis .......... the directory of LaTeX source codes of the thesis
text
  thesis.pdf ...... the thesis text in PDF format
case-study
  as-is ........... the directory with files, tables, models corresponding to as-is analysis
  to-be ........... the directory with models corresponding to to-be analysis
  implementation .. the directory containing implementation
  testing ......... the directory with results of testing
  screenshots ..... the directory containing screenshots of resulting system
```