



# Analysis and evaluation of modelling languages and workflow engines

Literature review in

Transportsystem management B.Sc. at  
Institute of Ubiquitous Mobilitysystems  
Faculty Informationmanagement and Media  
University of Applied Science

written by

**David Adam and Patrick Schuster**

Student number:  
58126, 57562

Supervisor:

B. Sc. Jonas Hansert

Leading Supervisor:

Dipl. Ing. Lucia Mejia Dorantes

Submission date: 20th January 2020



# Contents

---

<b>List of Abbreviations</b>	<b>IV</b>
<b>List of Figures</b>	<b>V</b>
<b>List of Tables</b>	<b>VI</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Background and principles . . . . .	1
<b>2 Modeling languages</b>	<b>3</b>
2.1 Structural perspective . . . . .	4
2.1.1 Entity-relationship model . . . . .	4
2.1.2 Class diagram (UML) . . . . .	5
2.2 Functional perspective . . . . .	7
2.2.1 Event-driven process chain . . . . .	7
2.2.2 Business Process Model and Notation . . . . .	9
2.3 Behavioral perspective . . . . .	11
2.3.1 Petri net . . . . .	11
2.3.2 State diagram (UML) . . . . .	13
<b>3 Workflow engines</b>	<b>15</b>
3.1 jBPM . . . . .	15
3.2 Camunda . . . . .	18
<b>4 Conclusion</b>	<b>21</b>
<b>Bibliography</b>	<b>VII</b>

# List of Abbreviations

---

<b>BPMN</b>	Business Process Modelling Notation.	9
<b>BPMN 2.0</b>	Business Process Modelling Notation 2.0.	18
<b>CMMN</b>	Case Management Model and Notation.	18
<b>DevOps</b>	Development and IT Operation.	19
<b>DMN</b>	Decision Management Notation.	18
<b>EPC</b>	Event - driven process chain.	7
<b>ERM</b>	Entity Relationship Model.	4
<b>IVR</b>	Interactive Voice Response.	17
<b>jPDL</b>	jBPM Process Definiton Language.	15
<b>KPI</b>	Key Performance Indicator.	19
<b>MSA</b>	Microservice Architecture.	19
<b>OMG</b>	Object Managment Group.	13
<b>REST</b>	Representational State Transfer pattern.	18
<b>SAFe</b>	Scaled Agile Framework for.	19
<b>SOAP</b>	Simple Object Access Protocol.	18
<b>UML</b>	Unified modelling language.	5
<b>WS-BPEL</b>	Webservice-Business Process Execution Language.	16
<b>XML</b>	Extended Markup Language.	18

# List of Figures

---

1.1	Abstracted difference between system and system context. [AMM <sup>+</sup> 19]	2
2.1	Model-based documentation divided into three perspectives.	3
2.2	Example of an ERM with rental bike scenario.	5
2.3	Example of an UML Classdiagram with rental bike scenario.	6
2.4	Example of EPC with buying a ticket scenario.	8
2.5	A ticket order process realised in a two lane pool by BPMN.	10
2.6	A navigation system realised as a Petri net.	12
2.7	State diagram with navigation device scenario.	14
3.1	Overview jBPM components. [Par93]	16
3.2	basic flow of the JBPM workflow engine. [Hui18]	16
3.3	Business transformation by using Process Automation Manager.	17
3.4	Overview Camunda components. [Cam19]	18
3.5	MSA of implemented solution with Camunda engines. [Fri]	20

# List of Tables

---

2.1	Tabular arrangement of Entity-relationship model elements. . . . .	4
2.2	Tabular arrangement of UML model elements. . . . .	6
2.3	Tabular arrangement of EPC model elements. . . . .	8
2.4	Tabular arrangement of BPMN elements. . . . .	9
2.5	Tabular arrangement of Petri net elements. . . . .	11
2.6	Tabular arrangement of UML Statediagram elements. . . . .	13
4.1	Comparison of the workflow engines Camunda and jBPM. . . . .	21

# Introduction

---

## 1.1 Motivation

In an increasingly information-driven and connected world, we are moving away from simple contexts and clear plausibilities. At the same time, the degree of complexity in a multitude of systems and technical realizations is increasing, making it the task of engineers to document these complexities in an abstract representation. Meaning, Models can transform the complexity of any system into a simple mostly graphic representation. Models are very important for todays software development, in fact they enhance the whole process of documenting and developing on large scope of software products because models help developers to understand the complexity behind the software.

In [Bra03], Bran Selic describes advantages of models in the software development process. More precise, he says that models are implementation and technology indepentend, closer to the problem domain and they enable the generation of artifacts like program code from models.

## 1.2 Background and principles

Recapping the independent attribute of models and todays complex environment, models need to indicate the difference between irrelevant and relevant objects. Therefore engineers mainly distinguish in system and system context while the irrelevant environment keeps objects which are not considered in the system architecture (Figure 1.1).

Borders between system and system context need to be defined for a better grasp of a system architecture. Requirements outside of the context border shall not be considered in the building process which allows a development process to be more stable, time and cost efficient.

A system context analysis examines relevant stakeholder, processes, systems, documents and events that have greater impact on the behaviour of the system. The system context represents linked aspects to the system, like subsystems, stakeholder and events by being directly linked through interfaces at the system border to the systems itself. [PR15]

According to [BS06] models follow fundamental principles, the main principles are likely to summarize in:

- **Abstraction**

Abstraction is a natural way of describing and classifying use cases. But objects in a model can be defined in a precise way too which leads often to an interplay between abstraction and concretion of a system.

- **Partitioning**

As already mentioned by the definition of system context, Partitioning considers sub-systems. The core aspect of partitioning is modelling *abstracted* interfaces and objects before to define precisely the individual structure.

- **Projection**

Misguided projection can lead to incompleted models which turns into insufficient and unsustainable system development. Projection takes different points of view into consideration, especially those of the different type of users and stakeholders. For example: The developer has a different view of the system than someone from management whose biggest satisfaction lies in user experience and covering cost-efficient requirements.

- **Modelling languages**

Modelling languages cover a wide range of deployment which is guided by standardisation e.g. ISO, rules and symbols. This lecture review will concentrate on main modelling languages whose can be applied into the phases of software development and give a fundamental understanding of model-based software engineering.

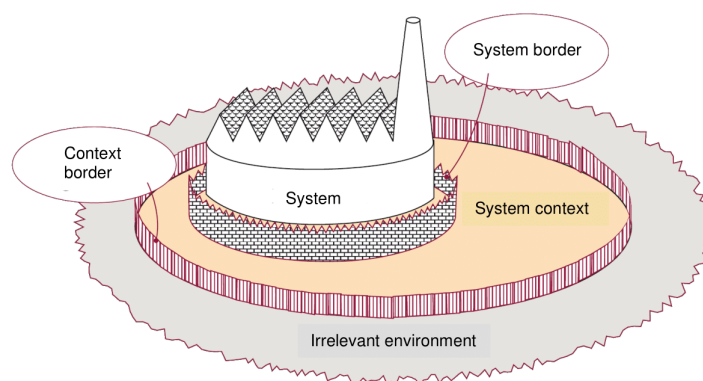


Figure 1.1: Abstracted difference between system and system context. [AMM<sup>+</sup>19]



# Modeling languages

A model is an abstract image of an existing or yet to be created reality [Sta73].

Besides natural language, system-requirements are often specified and documented by modelling languages. In general, experts map their functional system-requirements into three perspectives; Behaviour-, structure- and functional perspective (Figure 2.1).

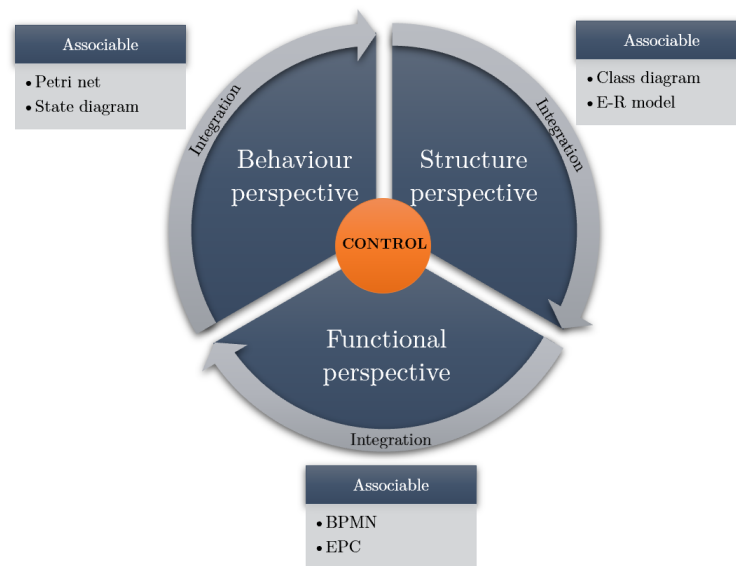


Figure 2.1: Model-based documentation divided into three perspectives.

Besides that, the illustration provides for each perspective an exemplarily set of modelling languages. For example, static data is modelled in class diagrams, object flow activities are found in the behaviour perspective as state diagrams and typically represent a functional perspective. [PR15]

Model-based requirements help to apply artefacts in a cross-functional team based on hard-level of formalisations and specified semantic for integration patterns.

## 2.1 Structural perspective

The structure perspective is a relevant modelling approach for the communication of mostly static-data and structure between systems and subsystems.

### 2.1.1 Entity-relationship model

The data modelling is an essential point in software development. A entity-relationship model (ERM) abstracts the structure and establishes semantics between systems and databases. Therefore, this language focuses on the abstraction of databases modelled in relational, object-oriented and networked data models. ERM is characterized by modelling static views of use cases and data, which does not mean that requirements towards process cycles or transformations can be ignored. Instead, the engineer is required to use modeling languages according to their areas of application and capabilities [BS06].

Following the principle of *abstraction*, ERM proposes two main elements (Table 2.1) to realize abstracted and highly acceptable models. The low complexity of the realization helps to present solutions in front of all types of stakeholder that are involved in the product life cycle.



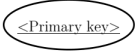

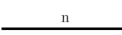
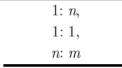
Elements	Symbol	Description
<b>Entity</b>		An <i>entity</i> describes an object that is independent of other entities and has a clear identifiable key attribute.
Attribute		Attributes specify the properties of an entity, but also the properties of relationships that do not belong to any of the linked entities. Refined into key attributes.
Key attribute		<i>Key attributes</i> are an attribute or a combination of attributes that uniquely identifies exactly one instance of an entity.
<b>Relationship</b>		In a semantic way, the entities are described in their relationship to each other.
Degree		The degree of relationship distinguishes the set of entities connected by a relationship. They are created by including objects in a recommended amount $n \leq 3$ .
Cardinality		<p>1: <math>n</math></p> <p>This cardinality describes that an entity type of type-1 may have zero or more relationships to entity type-2 (<math>n</math>), but not vice versa.</p> <p>1: 1</p> <p>This notation describes that an entity type of type-1 be related to only one other entity type of type-2.</p> <p><math>n: m</math></p> <p>This notation describes that an entity type of type-1 may have zero or more relationships to entity type-2, so entities of type-2 may also have more relationships.</p>

Table 2.1: Tabular arrangement of Entity-relationship model elements.

ERM provides abstracted information about static data between a set of entities. Figure 2.2 applies common ERM symbols to model a use case of booking a rental bike. The example is

divided by three entity types, two relationships and their individual attributes whereas the attributes of "ShareBike" is substantiated in a database table. Moreover, the table "ShareBikeGrid" describes and points out that relationship types and entity types are differentiated to their very own instances (so-called tuples).

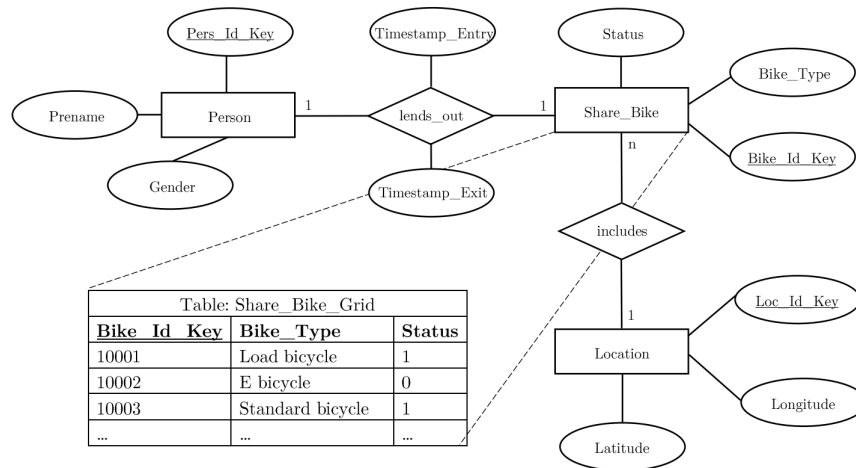


Figure 2.2: Example of an ERM with rental bike scenario.

### 2.1.2 Class diagram (UML)

Founded in 1997 by the Object Management Group, Unified Modeling Language (UML) has established itself as one of the most widely used modeling languages. [Ed 03] Basically, modeling with UML can be divided into two types. Static modeling is performed by the class diagram, while dynamic modeling is performed by sequence diagrams or state diagrams. In this chapter the most important parts of the class diagram are shown, which describes the structural perspective of the system. In particular, the UML class diagram is frequently used to describe the structure of a software system and thus forms the first basis for object-oriented modeling. The UML Classdiagramm combines the ideas of the Entity-Relationship modeling and the graphic representation of modules [BJW16]. Consequently the class diagram achieves a greater cardinality in the description of objects, for example classes [PR15].

Over the years, the ERM has become standard for the description of database models.[Sha92] However, according to the results from ([And08], [And98]), UML significantly simplifies the understanding of data models but has deficits when it comes to deduce properties about it. In addition, the use of UML class diagrams is less effective when it comes to specific questions about the relationships between two objects. Nevertheless UML is beneficial for software engineering. Vargas et al [Rut12] found that “at the system level, the change proneness of code modeled using class diagrams is lower than that of code that is not modeled at all.”

The following table (Table 2.2) shows the essential elements of a UML class diagram.

Elements	Symbol	Description
<b>Class</b>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> <div style="border-bottom: 1px solid black; margin-bottom: 2px;">&lt;&lt;stereotype&gt;&gt;</div> <div style="border-bottom: 1px solid black; margin-bottom: 2px;">&lt;Name&gt;</div> <div style="border-bottom: 1px solid black; margin-bottom: 2px;">Attributes</div> <div>Methods</div> </div>	<p>A class describes an object that is independent of other classes and has a clear identifiable name.</p> <p>Attributes and methods describe the class in more detail.</p> <p>A stereotype describes a class more precisely with regard to the type (interface, service, enum).</p>
<b>Association</b>	<div style="border-bottom: 1px solid black; width: 50px; margin-bottom: 2px;"></div> <div style="border-bottom: 1px solid black; width: 50px;"></div>	Association defines a general relationship between two elements.
<b>Generalization</b>	<div style="border-bottom: 1px solid black; width: 50px; margin-bottom: 2px;"></div> <div style="border-bottom: 1px solid black; width: 50px; position: relative;"><div style="position: absolute; right: -5px; top: -5px; border-left: 5px solid transparent; border-right: 5px solid transparent; border-bottom: 10px solid black;"></div></div>	Generalization maps a kind of inheritance. All methods, attributes and associations are inherited.
<b>Aggregation</b>	<div style="border-bottom: 1px solid black; width: 50px; margin-bottom: 2px;"></div> <div style="border-bottom: 1px solid black; width: 50px; position: relative;"><div style="position: absolute; right: -5px; top: -5px; border-left: 5px solid transparent; border-right: 5px solid transparent; border-bottom: 10px solid black;"></div></div>	Aggregation describes a type of relationship between two classes, where one class is part of another.
<b>Composition</b>	<div style="border-bottom: 1px solid black; width: 50px; margin-bottom: 2px;"></div> <div style="border-bottom: 1px solid black; width: 50px; position: relative;"><div style="position: absolute; right: -5px; top: -5px; border-left: 5px solid transparent; border-right: 5px solid transparent; border-bottom: 10px solid black;"></div></div>	Similar to aggregation, composition describes a part-whole relationship, but with more limitations. In the composition, the part cannot exist without the whole.
<b>Cardinality</b>	<div style="display: flex; justify-content: space-around;"> <div> <p>1: <i>n</i>,</p> <p>1: 1,</p> <p><i>n</i>: <i>m</i></p> </div> <div> <p>1: <i>n</i></p> <p>This cardinality describes that an entity type of type-1 may have zero or more relationships to entity type-2 (<i>n</i>), but not vice versa.</p> <p>1: 1</p> <p>This notation describes that an entity type of type-1 be related to only one other entity type of type-2.</p> <p><i>n</i>: <i>m</i></p> <p>This notation describes that an entity type of type-1 may have zero or more relationships to entity type-2, so entities of type-2 may also have more relationships.</p> </div> </div>	

Table 2.2: Tabular arrangement of UML model elements.

By transforming the example from the ERM, we get a more compact and understandable representation of the system in Figure 2.3. Following the model based approach we can directly derive domain objects from the model, due to the more exact description of the classes.

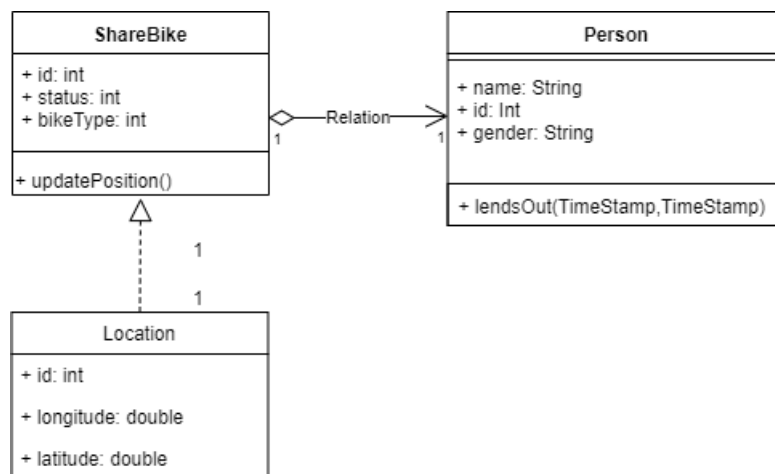


Figure 2.3: Example of an UML Classdiagram with rental bike scenario.

## 2.2 Functional perspective

In the functional perspective, the functionalities (which are available to the user) of a technical system are modelled in its system environment. Functional perspective considers the information in the system context, that is manipulated by a systematic event from the system.

### 2.2.1 Event-driven process chain

The Event-driven process chain is a modelling language used for describing business processes. EPC aims to describe processes at the business logic level and not necessarily at the formal specification level. [G. 92] Therefore, the strength of EPC modeling lies in its easy-to-understand notation, which makes it possible to map business information systems and simultaneously integrate properties such as functions, data, organizational structures and information resources.[BJW16]

One of the central deficits is that the semantics of an EPC are not clearly defined and it is also not possible to check the model for consistency and completeness. [Kro12] In contrast to structure modeling, process models always describe a dynamic view within an information model.[G. 92]

In a EPC, the process-related context is represented by functions. Functions are triggered by a starting-event. While events trigger the start of a function events can also be determined as a result of a function. There are a few important semantic restrictions to EPC, which are as follows.[Muh18]

- Events cannot make decisions like OR/XOR decisions. Events can only be linked with AND operator.
- Functions can be associated with all three logical operators (AND, OR, XOR) for decision making.
- Additional process objects can only be connected with the functions of EPC.

The following table shows the essential elements of an EPC model

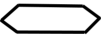





Elements	Symbol	Description
Event		An event represents a state that can trigger multiple functions.
Function		A function represents an action performed by a specific role. Functions can be associated with all logical operators.
Role		A role represents organizational unit which is performing the functions.
Data object		Stored or transmitted container for process-relevant information
System		Executing IT-Component.
Logical Operators		Logical operators (XOR, OR, AND)

Table 2.3: Tabular arrangement of EPC model elements.

Figure 2.4 shows the example model "buying bus ticket" with the EPC. The model clarifies which action is made by the user performing with the system and which action is made by the system itself. The first function is triggered by the user's desire to travel. The following steps depend on whether the user finds a destination or not. When he has located a destination for his trip, the system will show a number of ticket and payment options. The last action, performed by the user, requires the entry of money or credit card, so that the system can finally print the ticket.

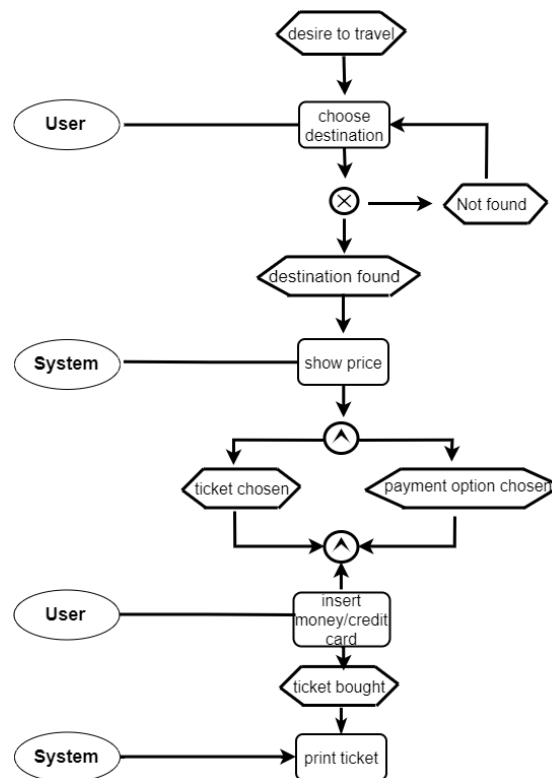


Figure 2.4: Example of EPC with buying a ticket scenario.

## 2.2.2 Business Process Model and Notation

Before we approach the key point of this chapter, it is important to first understand the basic term of Business Process Management in more detail according to 'BPM Common Body of Knowledge' [EUR09]:

"Business Process Management is a systematic approach for capturing, designing, executing, documenting, measuring, monitoring and controlling both automated and non-automated processes in order to sustainably achieve the goals aligned with the corporate strategy. BPM comprises the conscious and increasingly IT-supported determination, Improvement, innovation and maintenance of end-to-end processes."

Engineers often criticize the Business Process Model and Notation (BPMN) that it cannot be related to structures such as IT landscape, data and strategies but therefore other forms of notations exist. While this is the case, BPMN should be applied towards process design and chronological sequence of activities. According to the book [Jak16], the value chain is defined by processes that are represented by a collection of basic elements (Table 2.3).

Elements	Symbol	Description
Pool & Lanes	<div> <div>&lt;Pool&gt;</div> <div> <div>&lt;Lane1&gt;</div> <div>&lt;Lane2&gt;</div> </div> </div>	A <i>pool</i> is the environment that provides space to lanes and separates organizational boundaries. Where the <i>lanes</i> consists of entities and parts of the modelled process.
Event, Task	<div> <div>&lt;Task&gt;</div> </div>	Tasks are executed by participants along and to finalize a process. A task is defined by using verbs (prepare, build, send).
Subprocess	<div> <div>&lt;Subprocess&gt;</div> <div>+</div> </div>	In case a process is to complex and involves many steps, subprocesses are considered as solution to clearly structure complex processes.
Events		
Start-event		A start-event triggers the beginning of every process and is determined by a sequence-flow and task or gateway.
Final-event		A final-event shows the aiming target of an process. If all tasks are finished in the given sequence the event should close the process.
Interevent		An interevent is triggered while resolving tasks in a process and can have great impact on the following process execution.
Connectors		
Sequence-flows		Sequence-flows connect events and gateways in order to determine sequence of a process.
Message-flows		Message-flows are used for communication across pools therefore a precise description is needed.
Association		Associations connect additional participants, artefacts and IT-systems (Line to line)
Gateways		
XOR		Exclusive gateway will be applied If only one condition should be accepted.
AND		Parallel gateway activates all outgoing links. Following task waits for all links to be activated.
OR		Inclusive gateway activates one or more links that will be synchronized at the merging point.
Event-based		Event gateways are triggered by an event. Similar to the exclusive gateway, the process follows only one path
Artifacts		Artefacts keep information that is produced by events or related to events.

Table 2.4: Tabular arrangement of BPMN elements.

Numerous documentation [Dir] and [Ger] indicates that BPMN leads back to 2004 where the Business Process Management Initiative formalised this notation as a young standard to standardise interfaces of BPMS. The specification of BPMN is mainly focused on semi-formal characters and described in ISO/IEC 19510:2013. If we start to look from a global perspective towards BPMN, it is broadly acknowledged in the Anglo American areas that leads to pressure on EPC as a German formalised standard. Especially advantages like extensibility, separation of control-/information flow in different lanes and a more formalised specification are advantages of BPMN.

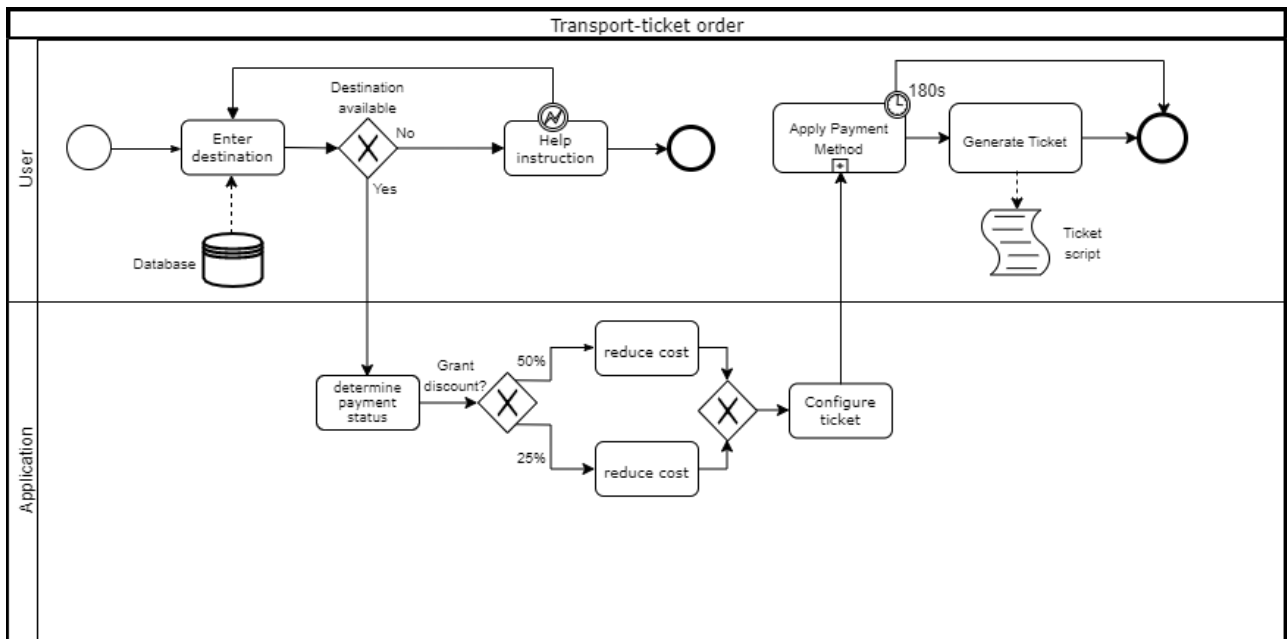


Figure 2.5: A ticket order process realised in a two lane pool by BPMN.

Above we look at a model for rising a transport ticket on a mobile application (Figure 2.5). As demonstrated by table 2.4, BPMN separates the user environment from the machine environment in different lanes to support a clear understanding of roles and organizational entities that are included within the process.

Based on the prerequisite that a user is logged into the system before the actual process has been started the user lane is in control of activities like input, storage and a integrated subprocess payment method. First, the user input will be validated where the input is handled by an exclusive gateway and can be either true or false.

Now the process is handled by the application to consider special services and create an artefact. In the final step the task 'Apply payment method' is realised as a subprocess and is followed by the task 'Save ticket'.



## 2.3 Behavioral perspective

The final section considers modelling approaches with a behavioural perspective towards the system and describes embedments of a system in its system context. The outcome is a model that leads to understand the behaviour after events, condition changes or even effects coming from the system context towards the system.[PR15]

### 2.3.1 Petri net

The basic model of the Petri net was introduced in the dissertation of Carl Adam Petri in 1962 [Pet62] and was subsequently standardised in [ISO04]. With its properties as a directed graph and disjunctive nodes (places and transitions) Petri nets allow graphical modelling and process simulation.

As a formal model for process modelling, the Petri net also has generic patterns following a semantic (table 2.5) in addition to the nodes. In the context of behavioural perspective state diagrams often appear as well. State diagrams use edges for transition modelling, while transitions in Petri nets belong to nodes. [WRGB13]





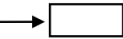
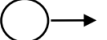


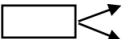
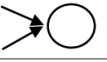
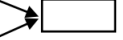
Elements	Symbol	Description
Transition, Event		This node represents occurring events and is used to model a moment within a time span.
Place, Action		The circular node type is used in modelling actions or conditions that have a time span.
Token/s		Tokens are a key aspect in petri nets modelling and only available in points.
Patterns		
Token creation		Generating tokens (Data, objects) <i>Passenger enters station</i>
Token erasure		Erasing tokens (Data, objects) <i>Passenger leaves station</i>
Storage (Source)		Storage for tokens as source (Data, objects) <i>Amount of vehicles at traffic control signal</i>
Storage (Archive)		Storage for tokens as archive (Data, objects) <i>Distribution of vehicles in different parking areas</i>
Excluding alternative		Excluding continues alternative <i>Vehicle takes turn at intersection or not</i>
Parallel action start		Parallel action start <i>Train arrives, visual at display and tone appears</i>
Asynchronous merge		Asynchronous merge more parallel actions <i>Central hub, multiple vehicle types come together</i>
Action Synchronization		Synchronization more parallel actions <i>Waiting passenger enters arrived Uber taxi</i>

Table 2.5: Tabular arrangement of Petri net elements.

Tokens describe the capacity in the places and can be used for workflow modelling (time span). The capacity description shows the amount of storable tokens that can move over edges between the nodes, but is not permitted to violate the specified edge weight.

The transition is used for switching regulation and mapping events (moment in time span). One of the requirements for operations (AND, XOR) of a circuit is that previous digits fulfil the minimum quantity of edge weights and that there is sufficient residual capacity in the next place. Based on the edge weight, the circuit removes the required quantity of tokens from the previous place. [BS06]

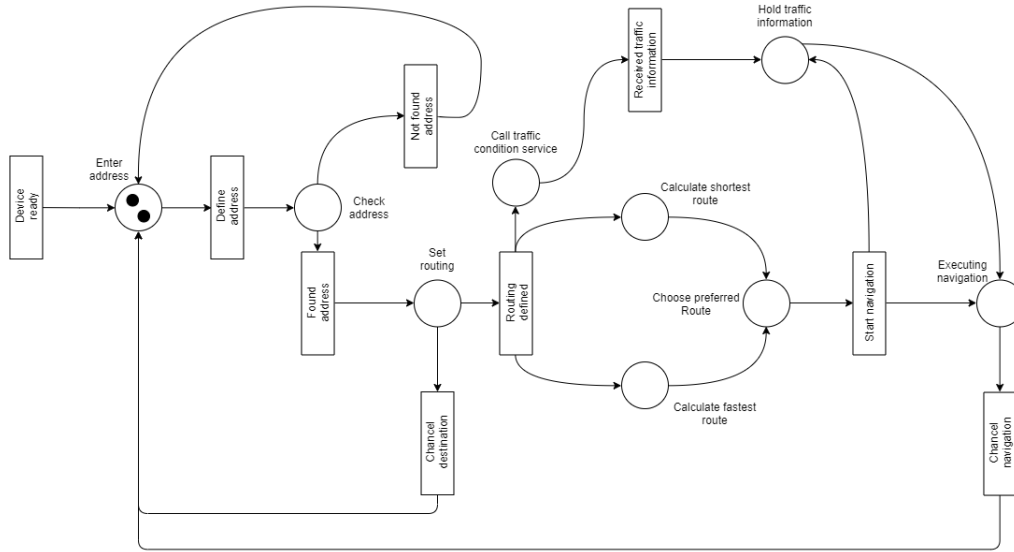


Figure 2.6: A navigation system realised as a Petri net.

An illustration of a possible Petri net is shown in figure 2.6. It takes the use case of starting a route in a navigation system on a vehicle. This model can be seen as an active Petri net due its condition 'Enter address' keeps two tokens. Beyond that it is triggered by having an event 'Device ready'.

Passing the address check place for availability, a token either moves back to 'Enter address' or starts an activity 'Found address'. The model continuous as excluding alternative in 'Chancel Destination' or 'Routing defined'. If not chancelled, multiple parallel activities will start 'Call traffic condition service' and route calculations, whereas route calculation is despite into two main route options before starting the navigation. While routing, the second token initialises traffic data service and holds this data in another action place. By the time the navigation starts the execution is provided by traffic data. In case of a chancelled navigation the system prompts to enter a new address.

### 2.3.2 State diagram (UML)

Based on the notation of Statecharts developed by [Dav87], the OMG specified the model elements of the UML Statediagram in 2007. Both Statecharts and UML Statediagrams describe a technique to model the behavior of a system based on the concept of finite state machines. By definition, a finite state machine comprise a set of states and state transitions which, depending on the current state of the state machine, are executed by the occurrence of an event. A state defines a period of time in which the system under consideration shows a certain behavior and waits for the occurrence of a defined event. The change from one state to the next is triggered by events and is called state transition. Depending on the intended use, hierarchization in state machines allows to form superstates and to abstract from the possibly complex behavior in these states. [PR15]





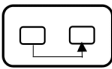
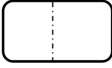
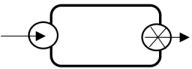
Elements	Symbol	Description
State		A state defines a period of time in which the system under consideration shows a certain behavior and waits for the occurrence of a defined event.
Start-State		Starting State of the system.
Final-State		Final State of the system. There can me more than one Final State.
Transitions		Represents a state transition.
Hierarchization		Statecharts enable a hierarchy of states by describing a state again by an automaton
Parallel Statechart		Parallel automaton can be defined with conditions for state transitions
Entry/Exit Point		An external visible pseudo state.

Table 2.6: Tabular arrangement of UML Statediagram elements.

The UML 2 extends the model elements of statecharts by the possibility to define explicit entry and exit points for hierarchical states. [RJB99] describe an entry point as an externally visible pseudo state that is directly associated with an internal state. An exit point as an externally visible pseudo state that has an internal state as its origin. A superstate within a state machine can have any number of entry and exit points defined by a name.

Figure 2.7 shows a modelled scenario of a navigation device. The initial state is when the navigation device is ready. The next State Transition is triggered by the users decision whether he wants to be navigated to a new destination or the last destination set in the system. Both decision end in the superstate "Route calculation". There is a loop between "Route calculation" and "Display Route", because the Route has to be permanatly updated. The final state and at the same time the initial state is reached when navigation is aborted or successfully completed.

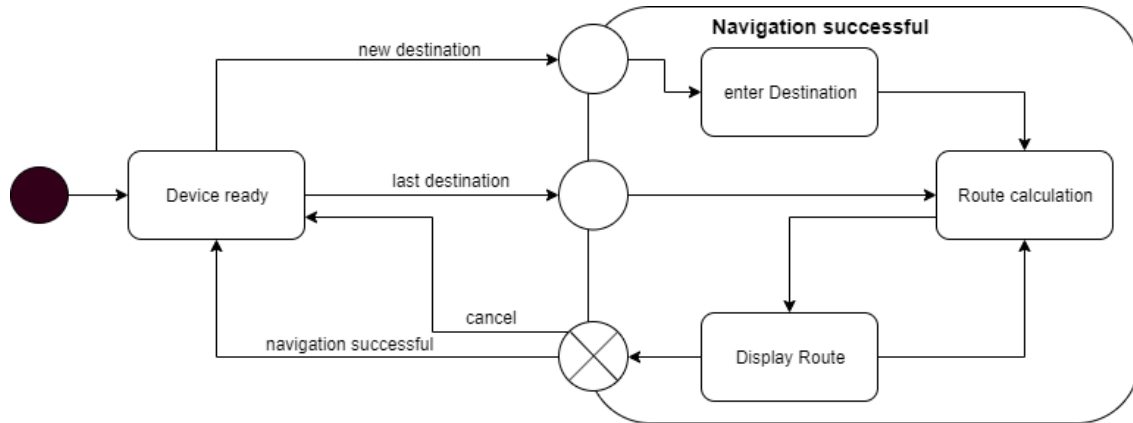


Figure 2.7: State diagram with navigation device scenario.

# Workflow engines

---

## 3.1 jBPM

Java Business Process Management is a free open source workflow management framework developed by JBoss, based on Java language. jBPM is an lightweight, open source, flexible, easily extensible executable process language framework that covers business process management, workflow, and service collaboration. jBPM mainly includes Workflow Engine, flow monitoring tool and Graph Process Designer based on Eclipse platform. [Hui18]. [Par93]

### 1. Workflow Engine

The workflow engine controls the runtime process, provides interfaces for process definition, process management and process monitoring and user interfaces through which systems can interact with the workflow engine. As shown in Figure 3.1 It is designed with a layered architecture. The engine consists of the four core classes Execution Service, Task Service, History Service and Identity Service [Hui18]

### 2. Process Designer

This is an Eclipse plug-in, which provides support for defining processes in jPDL both in a graphical format and in XML. jPDLs is the process language used by the system. [TN10]

### 3. Workflow Client

It is a web-based client through which users can initiate processes and execute activities through an API. The user interface also allows administrators to monitor processes and instances. Furthermore, the Identity component, takes care of the definition of organizational information, such as users, groups, and roles.[TN10]

jBPM enables flexibility by supporting multiple-process languages with the same scalable process engine platform. jBPM not only supports jPDL but also other XML-based process languages like: WS-BPEL and Seam Pageflow. [Zha08]

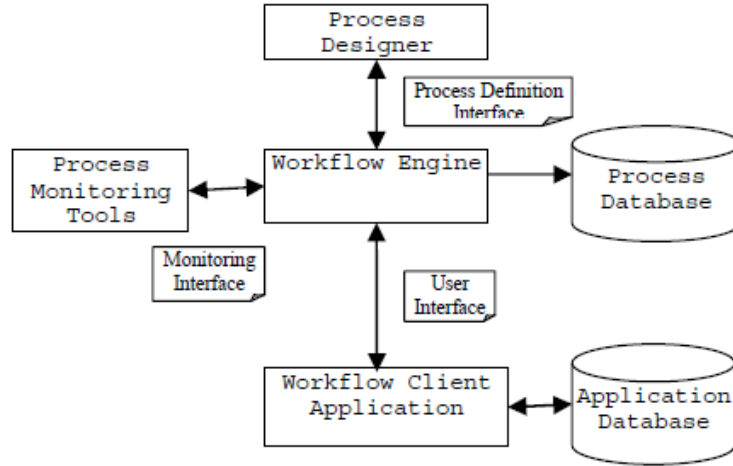


Figure 3.1: Overview jBPM components. [Par93]

According to [Hui18] a basic flow of jBPM workflow engine consists of the components shown in Figure 3.2. First define the process that meets the needed business requirements. When the process is defined, use the graphical process designer to draw the business process. It is possible to use the designer to generate the jPDL or directly write the XML document corresponding to the jPDL. The jBPM workflow engine creates process instances depending on the underlying jPDL or XML. Then, according to a specific business, a business document report can be created. Users can invoke the relevant interfaces provided by jBPM to realize the specific process operations of the enterprise through the system interface.

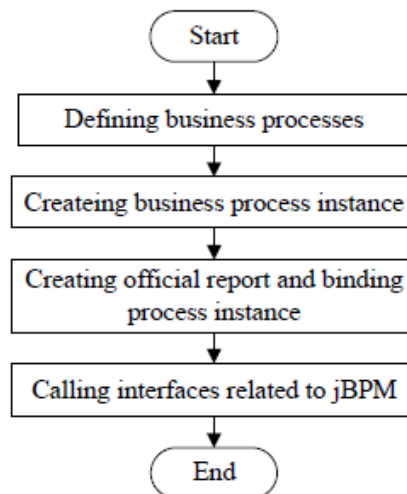


Figure 3.2: basic flow of the JBPM workflow engine. [Hui18]

Using the fundamental engine and modal behind jBPM, Red Hat developed a Process Automation Manager which comes with a higher performance. The Process Automation Manager is a platform for developing containerized microservices and applications that automate business decisions and processes. [Red19] The Laser Group, in partnership with Red Hat, introduced the Process Automation Manager to streamline business processes. Figure 3.3 shows the transformed business process by using the Process Automation Manager. The IVR now sends all information to the Process Engine, which addresses the respective responsible service.

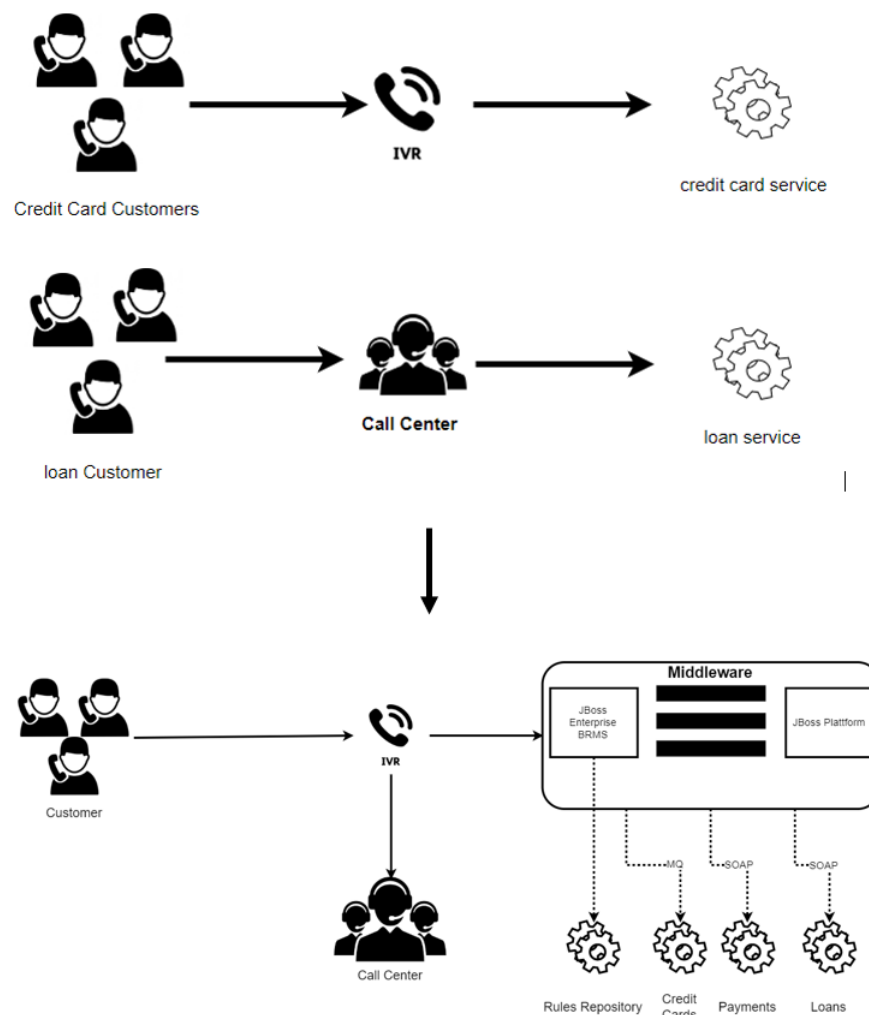


Figure 3.3: Business transformation by using Process Automation Manager.

## 3.2 Camunda

The Camunda stack provides a toolset of execution and decision engines for BPM workflows and process modelling so that projects can be structured, automated and executed. Enterprises and institutions of various industries like corporate, startups and public with medium to large size, a complex IT landscape and strives for automated processes benefit from Camundas components. [Rob19]

The Java-based framework orchestrates and supports agile and common open source services to enable a better work environment for various team roles: System architects, software engineers, project manager and business analysts. Figure 3.3 illustrates the Camunda infrastructure and components together with typical team roles (developer, manager, analyst), implemented standards (BPMN, DMN) and architectural functionalities (container-managed Java engine). [Cam19]

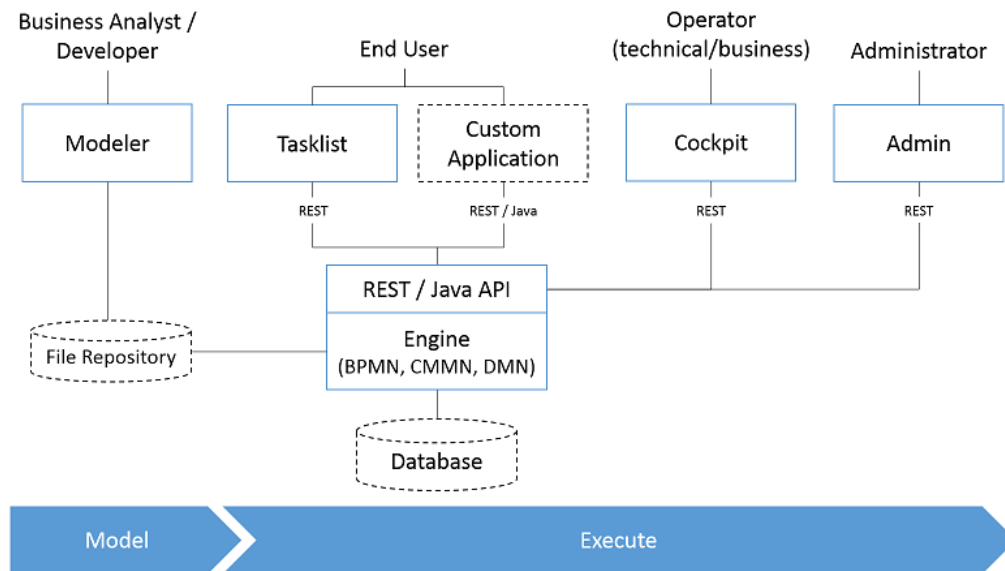


Figure 3.4: Overview Camunda components. [Cam19]

Camunda Modeler is a high-level modelling tool based on BPMN 2.0, CMMN and DMN standards that embraces analysts and developers to design processes given by projects or enterprise strategies. An integrated SOAP and REST service enables a local repository to bridge deployments between the Camunda Modeler (XML formatted process) and the executing engine. Camunda has a lightweight engine that is responsible to execute given tasks and decisions. In addition, Camunda provides an open source REST interface for requesting web applications. Provided or self-customized web applications provide components that manage workflow tasks (Tasklist), process monitoring and error handling (Cockpit) and authorization (Admin) functionalities. [Cam19]



Best practices of Camunda can be found in IT solutions with any project context in order to initiate, plan, control and evaluate individual process automation or detect system related gaps. This section focuses on the Camunda integration towards stages of a technology-driven project. [Rob19]

### 1. Project Scoping

Views on project scoping range from project objectives to practical activities to define costs of errors and manual work, quality of transparency, throughput, time to market and adaptations. Camunda is used to give a pre-study about project volume and risks which is a certain need in the stage of proofing a concept. Equally important is communication between team roles that negotiate project vision by BPMN and DMN.

### 2. Design and Implementation

Next stage is design and implementation to accumulate functional and non-functional requirements in an incrementally approach. Besides that, the team roles and tasks are defined and introduced to domain knowledge. Camunda will be technically integrated into consisting architecture. Whereas existing instances of processes, decisions and optimisations are modelled and maintained against performance issues.

### 3. Solution Rollout

A solution rollout expects a closed test cycle and acceptance by management before released to operations. Now, release and change management take over and ensure security, version control and monitoring enabled by Camunda Cockpit.

### 4. Continuous Improvement

As result the automated process can be continuously improved by analysing operational data, KPI monitoring and error handling. Camundas retrospectively function helps to identify enhancements, weak points and capacity planning of affected components.

Looking at the case study of Deutsche Telekom [Fri], a German telecommunications company, Camunda has been partially applied to break down a system from monolytical architecture into a microservice based architecture. Historically, the issue focussed on a inflexible IT landscape of a TAL-order process back in 2008 that shall be reengineered for fiber-optic communication by a new service oriented architecture. In 2018, leading engineers identified 4 main drivers to this project: *MSA*, *SAFe* to enable bottom up and top down workflow, *Cloud* technology reduces maintainance, provides scalability and self service and *DevOps* to manage complex application stacks.

Following the microservice architecture (MSA) vision, the existing monolith partitions into self-sustaining and lightweighted services. This was achieved by categorise into frontend-/ business-/ backend data, group functionalities to ensure consistent order processing, data calculations and finally, define and assign three microservice categories: Business process microservice (incl. Camunda), data microservice and domain microservice (incl. Camunda).

The meshed solution is provided by two independent coordination patterns of services. Unfortunately, a orchestrated solution was not suitable due its centralised BPM engine. Furthermore, a choreography oriented solution has had advantages in its loose coupling but disadvantages in locating processes and orders. To conclude, the final solution is a 'choreographed orchestration' which enables each microservice component to keep a BPM engine while communicating with a messaging broker. [[Cam17], [Fri]]

These factors contribute to a more performant architecture with lightweighted and Java-based development, supports exception handling in operations and influences IT-Business alignment by visualizing complex logic.

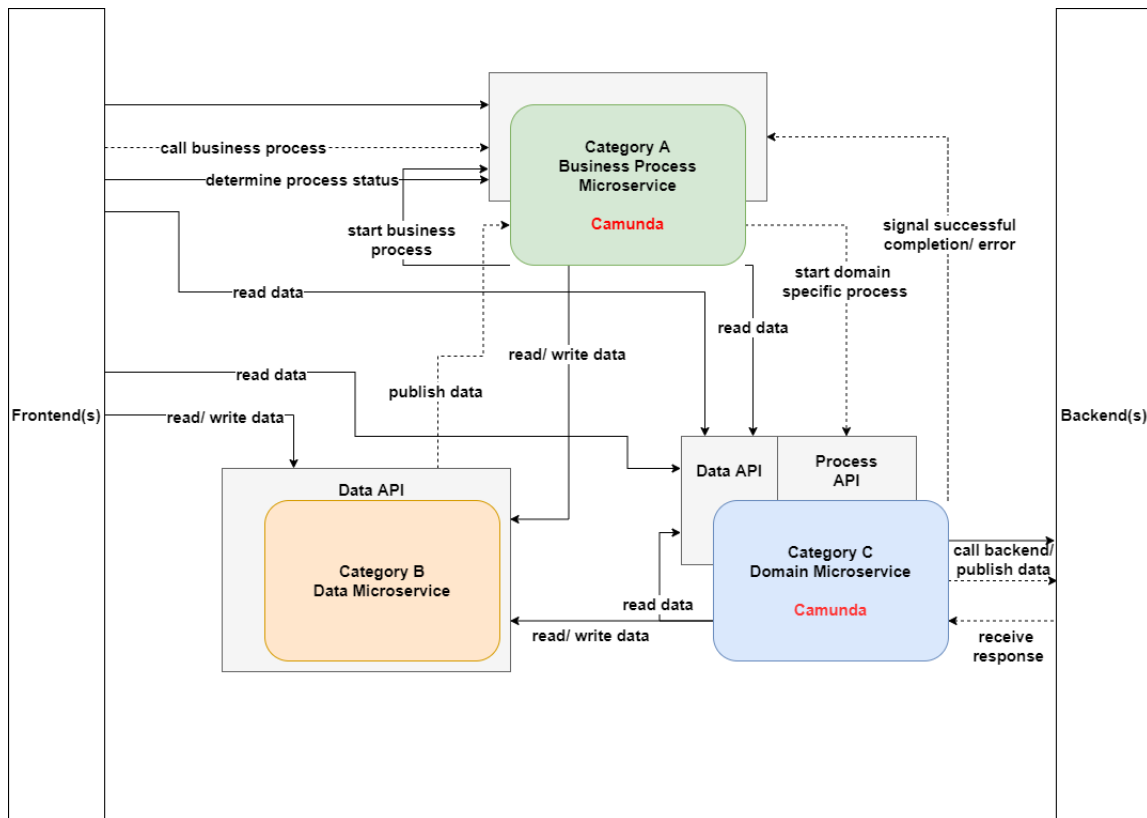


Figure 3.5: MSA of implemented solution with Camunda engines. [Fri]

# Conclusion

---

The different perspectives show that, depending on the use case, it is possible to choose from a pool of modelling languages. When it comes to the structural perspective, the use of UML class diagrams has proven to be a good choice. ER models are therefore no worse than UML class diagrams; they address a different domain. For example, they are consistent when it comes to lightweight modeling of relations. In the modeling of the functional perspective, we can observe that BPMN is used more in companies for business process modeling, since BPMN allows the modeling of more complex processes due to the specification. Furthermore, it is because there are workflow engines for BPMN. The presented modelling languages of the behavioural perspective behave similar to those of the structural perspective. Modelling with Petri nets has been done since 1960. The corresponding UML was specified much later, which is significantly reflected in the understanding of the language. When it comes to modeling workflows, Petri nets address active systems, rather than reactive ones. Furthermore Petri nets can be described mathematically, which makes them interesting for mathematical models.

The following table 4.1 illustrates a functional comparative of the workflow engines Camunda and jBPM. Both provide a graphical modeler, but do not offer the possibility to add functionality. Depending on the system context, developers decide which engine is more suitable for the use case.

	<b>Camunda</b>	<b>jBPM</b>
<b>BPMN 2.0</b>	Conform	Conform
<b>Prototyping</b>	Available but with support of experts	Available, but with support of experts
<b>API</b>	IT experts	IT experts and operations
<b>Editor</b>	Oriented to IT experts	Oriented to business analyst with/without IT experts
<b>Usability</b>	Outsourced IDE, web application, System application	Outsourced IDE, web application, System application
<b>Update of console application</b>	Not allowed	Not allowed
<b>Business process monitoring</b>	Oriented to IT experts	Oriented to business analysts and IT experts
<b>Update of source code</b>	Community support platform, availability of source code	Community support platform, availability of source code

Table 4.1: Comparison of the workflow engines Camunda and jBPM.

# Bibliography

---

- [AMM<sup>+</sup>19] Amelie Lang, Markus Wiener, Michael Bezold, Chris Rupp, and Jan Dovica (2019), Videos im re: Das sagemumwobene umgebungsvideo. URL <https://blog.sophist.de/2019/04/17/videos-im-re-das-sagemumwobene-umgebungsvideo/>.
- [And98] Andy S. Evans (1998) *Reasoning with UML Class Diagrams*. 2nd IEEE Workshop on Industrial Strength Formal Specification Techniques.
- [And08] Andrea De Lucia, Carmine Gravino, Rocco Oliveto, Genoveffa Tortora (2008) *Data Model Comprehension: An Empirical Comparison of ER and UML Class Diagrams*. 16th IEEE International Conference on Program Comprehension.
- [BJW16] Benker, T., Jürck, C., and Wolf, M. (2016) *Geschäftsprozessorientierte Systementwicklung: Von der Unternehmensarchitektur zum IT-System*. Springer Science and Business Media : Springer Vieweg, Wiesbaden.
- [Bra03] Bran Selic (2003) *The pragmatics of model-driven development*. IEEE Xplore.
- [BS06] Bernroider, E. and Stix, V. (2006) *Grundzüge der Modellierung: Anwendungen in der Softwareentwicklung*. Manual, WUV, Wien, second edition.
- [Cam17] Camunda Services GmbH (May, 2017) *Microservices and BPM*. Camunda GmbH, Berlin.
- [Cam19] Camunda Services GmbH (2019), Camunda docs - bpm platform. URL <https://docs.camunda.org/manual/7.7/>.
- [Dav87] David Harel (1987) *Statecharts: A visual formalism for complex systems*. Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel.
- [Dir] Dirk Stähler, Bpmn verdrängt die epk? URL [https://www.gbtec.de/de/ueber-gbtec/veranstaltungen/webinar\\_bpmn\\_vs\\_epk/](https://www.gbtec.de/de/ueber-gbtec/veranstaltungen/webinar_bpmn_vs_epk/).
- [Ed 03] Ed Seidewitz (2003) *What Models Mean*. IEEE Software.
- [EUR09] EUROPEAN ASSOCIATION OF BPM (2009) *Business process management common body of knowledge*. Association of Business Process Management Professionals, Indiana.
- [Fri] Friedbert Samland, W.T., Monolith to microservice, waterfall to agile - success with camunda.
- [G. 92] G. Keller, M. Nüttgens, A.-W. Scheer (Januar 1992) *Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten*.

- [Ger] Gero Decker, W.T. *Migration von EPK zu BPMN*.
- [Hui18] Hui Jiang (2018) *Research and Implementation of Financial Approval System Based on jBPM Engine(IAEAC)*. IEEE, Tianjin, China.
- [ISO04] ISO/IEC 15909-1 (12-2004), Systems and software engineering – high-level petri nets.
- [Jak16] Jakob Freund, B.R. (2016) *PRAXISHANDBUCH BPMN 2.0*. CARL HANSER Verlag GMBH.
- [Kro12] Krogstie, J. (2012) *Model-based development and evolution of information systems*. Springer, London.
- [Muh18] Muhammad Waseem Anwar (March 13, 2018) *Event-Driven Process Chain for Modeling and Verification of Business Requirements*. IEEE Access.
- [Par93] Paredes Villegas, A. (1993) *Research and Design of Document Flow Model Based on JBPM Workflow Engine*. Colección Ricardo Miró, Instituto Nacional de cultura Dirección Nacional de Extensión Cultural Departamento de Letras, Panamá, first edition.
- [Pet62] Petri, C.A. (1962) *Kommunikation mit Automaten*. Dissertation, Fachbereich Informatik, University Hamburg.
- [PR15] Pohl, K. and Rupp, C. (2015) *Basiswissen Requirements Engineering: Aus- und Weiterbildung zum "Certified Professional for Requirements Engineering"*. dpunkt-Verlag, Heidelberg, fourth edition.
- [Red19] Red Hat (2019), Red hat process automation manager. URL <https://www.redhat.com/de/technologies/jboss-middleware/process-automation-manager>.
- [RJB99] Rumbaugh, J., Jacobson, I., and Booch, G. (1999) *The unified modeling language reference manual*. The Addison-Wesley object technology series, Addison-Wesley, Reading, Mass. and Harlow.
- [Rob19] Robert Gimbel (20.2.2019), Camunda customer journey. URL <https://de.slideshare.net/camunda/camunda-bpm-in-dach-71590192>.
- [Rut12] Rut Torres Vargas, Ariadi Nugroho, Michel Chaudron, Joost Visser (2012) *The use of UML class diagrams and its effect on code change-proneness*. 23rd International Conference on MODELS, IEEE Software, Innsbruck, Austria.
- [Sha92] Shamkant B. Navathe (1992) *Evolution of data modeling for databases*. Communication of the ACM.
- [Sta73] Stachowiak, H. (1973) *General Modeltheorie*. Vienna Springer, Wien.
- [TN10] Ter Hofstede, A. and Nick Russell (2010) *Modern business process automation: YAWL and its support environment / edited by Arthur H.M. ter Hofstede ... [et al.]*. Springer, Heidelberg.

- 
- [WRGB13] Winter, M., Roßner, T., Götz, H., and Brandes, C. (2013) *Basiswissen modellbasierter Test*. dpunkt Verlag, first edition.
- [Zha08] Zhang, Y. (2008) *Research on workflow patterns based on jBPM & jPDL*. IEEE, Piscataway, NJ.