universität ulm

Design and Implementation of a Framework for the Creation of BPMN 2.0 Process Models based on Textual Descriptions

---

Master Thesis

Kai Kretschmann

903179
kai.kretschmann@uni-ulm.de

Institute of Databases and Information Systems
Faculty of Engineering, Computer Science and Psychology
Ulm University

Reviewer:
Prof. Dr. Manfred Reichert
Prof. Dr. Rüdiger Pryss

Supervisor:
Michael Winter

Submission:
14.December.2021

Acknowledgement

I want to thank Prof. Dr. Manfred Reichert and Prof. Dr. Rüdiger Pryss for reviewing my work. My sincere thanks goes to Michael Scheurer for the best advices and help he provided, I also want to thank Nina Beranek, Maximilian Blasi, Patrick Gröger, Hannah Lappe, Steffen Leger and any other friends for their help.

Furthermore, I want to thank my supervisor Michael Winter for his advice, guidance and the possibility to work on this topic.

Abstract

There exists a plethora of different modeling tools for Business Process Model and Notation (BPMN) 2.0, most of these tools do not combine modeling with textual description. But often textual descriptions of a process are provided, this description has to be analyzed and later implemented as a process model. Especially for beginners this is a difficult task and as they first need to focus on finding the important elements in the text and later create the model with this information. Performing these two tasks in conjunction should ease the process of modeling BPMN for beginners. For this, a web application was developed which combines the modeling and the textual description. During the concept phase psychological aspects were kept in mind to further support the user. It especially should help to lower the cognitive load and keep the motivation of beginners. This approach differs from the development of BPMN models from a textual description with natural language processing (NLP) because the application is still focused on the manual creation of models by the modeler and unlike NLP approaches where models are created automatically. A reason for this approach and not using NLP for the creation of BPMN models is that it will always be important to have good process modelers, who can create models manually and can also analyze these processes, find weaknesses and improve them.

Contents

1 Introduction

1.1 Motivation

Today, through increasing globalization, more and more organizations have to optimize their structures and processes to keep themselves competitive. To do so they use business process models, which help to understand the structure of a process in an organization or are used as blueprints for process implementation [1]. Business process models are mostly simple to understand, even if you are no expert [1]. This means processes can be analyzed even by someone who does not already know the process. The processes can also be easily improved with the help of business process models because for example through these models weaknesses can be found. Due to these useful aspects and the growing complexity of organizations, business process models become more important. To refine the understanding of process models and because of the increasing demands more research was conducted in recent years [2]. Due to this research, it is now clear that there exists a knowledge gap between experts and novices in understanding business process models [2]. To close this knowledge gap it is important to help improve the learning process of novices. Thanks to different research on learning processes, nowadays we know that there are different aspects that influence the learning process: One influence among these is the cognitive load, another is the motivation of performing the learning task. This knowledge could be used to create modeling tools which help to improve the learning process of novices and is also useful for other knowledge levels. One business process modeling language is the Business Process Model and Notation (BPMN) 2.0, which is very common and used for example for analyzing of processes, Robot Process Automation and Robot Desktop Automation. A common method of learning BPMN is, to have a process description and to try to create the corresponding process model. This makes it hard for novices to start because they have to understand the description at the same time while modeling, which increases the complexity of the task.

Because of this, this thesis focuses on the development of a web application that helps novices to increase their modeling skill with BPMN and improve their learning process. To achieve this, the psychological aspects which influence learning are kept in mind.

1.2 Objective

In this work, a web application which should improve the learning process of novices and help to create simple BPMN models should be created. For this purpose, different psychological aspects which have an influence on learning and help lead the user to the creation of good BPMN models were taken into account. Furthermore, it was important to keep the design and modeling different from other tools. An important goal was that the application can combine textual description and modeling to keep the complexity of creating process models low. This should help novices to increase their knowledge on how to understand a process description while creating a process model.

1.3 Structure

After the motivation and objective of the thesis, Chapter 2 introduces the concept of business process modeling and in Section 2.1 different business process modeling languages are described. Then Section 2.2 explains details of BPMN 2.0 and provides information on BPMN which is needed for the development of the application. After that Chapter 3 presents psychological aspects which are used in the conception phase of this work and have an influence on the learning process. First, Section 3.1 will explain the concept of nudging with some examples, knowledge which is used in the conception phase along with Natural Language Processing. The different types of motivation and what affects motivation is explained in Section 3.2. Subsequently, Section 3.3 will provide a view on cognitive load and how it influences learning. Section 3.4 describes the meaning of color harmonization and color emotion; this information is used to find the best fitting colors for the application. In Chapter 4 work which has similar aspects as this thesis is discussed, for example the use of natural language processing. Then in Chapter 5 requirements for the application are defined. These are separated into functional requirements in Section 5.1 and non-functional requirements in Section 5.2. In Chapter 6 the steps which were taken to get the final concept of the application and how the psychological aspects influenced this conception phase are explained. It is also explained how it

should be possible for the user to create a BPMN XML and how components of the application should interact during the creation process of the BPMN XML. After that in Chapter 7, the development of the application is explained. First, Section 7.1 explains which technologies were used and why they were used. In Section 7.2, the architecture of the application will be described. In Section 7.3 parts of the implementation of the application are explained. Subsequently, Section 7.4 states the major problems during the implementation process and describes how they were solved. In Chapter 8 a short demonstration of the application is provided. Then in Chapter 9 the requirements which were defined in Chapter 5 are investigated, which of them are met, need to be checked, or are not met. The thesis concludes with Chapter 10, which provides a summary and an outlook on possible future work.

2 Theoretical Background

In this Chapter the fundamental theoretical background needed to understand the application which is developed later is explained. First, it is described what business process modeling is and in Section 2.1 what are possible business process modeling languages. Thereafter, in Section 2.2 the fundamentals of Business Process and Notation (BPMN) 2.0 are introduced.

The simplest description of business process modeling is: Business process modeling is an approach on how organizations conduct business processes in the future and now [3]. With business process modeling organizations can improve their Business Process Management (BPM) more easily. Process models describe activities and events of business processes in a graphical way [3]. Created business process models can be analyzed simple, improve the efficiency and quality of the process for the future [4]. For more detailed information about business process modeling the work of Indulska et al. [3] and Mili et al. [5] can be interesting.

2.1 Business Process Modeling Languages

Now we have a look at different business process modeling languages. One of these languages is the Business Process Model and Notation from the Object Management Group (OMG) [6] which is discussed in detail in Section 2.2. Some modeling languages were developed with different intentions, but have been used to describe business processes. In the work of Mili et al. [5] they split these languages into four traditions as following [5]:

- *Traditional process modeling languages*: These languages are typically not formal. Languages of this category for example are Petri Nets and Event Process Chain (EPC).

- *Object-oriented languages*: These languages mostly represent the software domain, rather than the business domain. A well known example would be UML (Unified Modeling Language), which is most often used in software development.
- *Dynamic process modeling language*: Languages of this tradition are mostly proposed from various organizations. Example languages of this category are BPMN and Workflow Process Description Language (WPDL).
- *Process integration languages*: These languages typically were created to manage business overarching processes. Languages of this category are electronic business extensible markup language (ebXML) and Web Services Choreography Description Language (WS-CDL).

2.2 Business Process Model and Notation 2.0

In this Section, the fundamentals of BPMN 2.0 are described with extra details on elements which are essential for the development of the application.

BPMN is a standard for business modeling, the first version was officially published in 2006 by the Object Management Group (OMG) [7]. The latest version of 2.0 is version 2.0.2 which was published in January 2014 [8], the 2.0 version was published in December 2010 [6]. The OMG developed BPMN 2.0 with the intention that it is used directly by the stakeholders who handle the processes but also allows BPMN diagrams to be translated into software process components [8]. Nowadays, BPMN is often used for Robot Process Automation (RPA) and Robot Desktop Automation (RDA) because these two automation make it easier and faster to handle processes [9].

According to the specification of BPMN 2.0, there are different BPMN sub-model types. There are three different types specified, these types are [6]:

- **Processes** (orchestration), including: Private executable and non executable **Business Processes** and public **Processes**
- **Choreographies**
- **Collaboration**, which is a view of **Conversations** between **Processes** and/or **Choreographies**

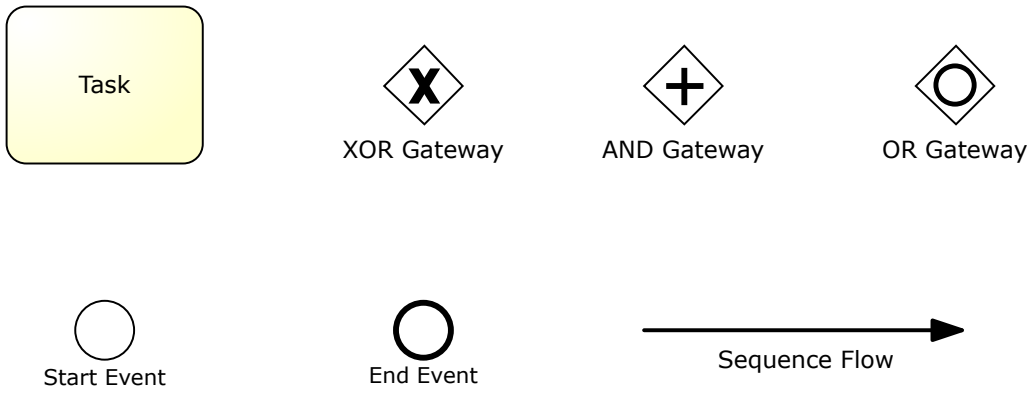

Figure 2.1: The basic elements from the BPMN 2.0 specification [6]

In order to keep it simple this work focuses on Processes, not on Collaboration and multi-participant elements will be omitted. Now a look on the basic elements of BPMN 2.0 which are needed for the later use in this work. These basic elements can be seen in Figure 2.1.

The elements are explained as follows in the specification [6]:

- **Task**: A Task is an Activity, which is not broken down into a more detailed level. An Activity is a term for work performed in a process.
- **Gateways**: Gateways are used to control the divergence and convergence of Sequences Flows. The internal markers indicate different behavior. They are usually used to visualize decisions or parallels of an process.
- **Start/End Event**: A Start Event implies where a process starts, an End Event indicates where a process ends.
- **Sequence Flow**: Sequence Flows are used to show the order of a process.

All of these basic elements have a representation within XML. They also need a unique ID, which is used as an identifier for defining relationships with other elements and is also used for the layout of the elements. The XML representation of a Start Event can be seen in Listing 1, corresponding to the Start Event is the End Event representation in Listing 2. The only difference between these two is that Start Events only have outgoing sequences and End Events only have incoming sequences. Tasks have a quite simple XML

```
<startEvent id="tag.id" name="name">
    <outgoing> sequenceID </outgoing>
</startEvent>
```

Listing 1: XML representation of a start event

```
<endEvent id="tag.id" name="name">
    <incoming> sequenceID </incoming>
</endEvent>
```

Listing 2: XML representation of an end event

representation, their representation can be seen in Listing 3; they have incoming and outgoing sequences. The XML representation of split and join, are shown in [Listing 4, Listing 5]. The difference between them is that joins have more incoming sequences while splits have more outgoing. Another difference is that the gateway direction for a split is diverging, and for a join, it is converging. Important for the different splits and joins is, whether another split or join instead of an XOR is needed, the Tag **exclusiveGateway** has to be changed to **parallelGateway** if an AND is needed, or to **inclusiveGateway** if an OR is needed. One of the most important elements are Sequence Flows, their XML representation is quite simple and is shown in Listing 6. The representation needs a unique ID, the ID of the predecessor element, and the ID of the successor element.

All of these XML representations can be found in the specification of BPMN [8] but for a correct BPMN XML the elements do not only need a process representation, they also need a diagram representation, which describes the positions of the elements, their size and waypoints for a visual display of the BPMN. For the positioning of elements it is important to know where their coordinate system starts. The coordinate system starts

```
<task id="tag.id" name="name">
    <incoming> sequenceID </incoming>
    <outgoing> sequenceID </outgoing>
</task>
```

Listing 3: XML representation of a task

```
<exclusiveGateway gatewayDirection="Diverging" id="tag.id"
name="name">
    <incoming> sequenceID </incoming>
    <outgoing> sequenceID </outgoing>
    <outgoing> sequenceID </outgoing>
</exclusiveGateway>
```

Listing 4: XML representation of an XOR-Split gateway with two outgoing sequences

```
<exclusiveGateway gatewayDirection="Converging" id="tag.id"
name="name">
    <incoming> sequenceID </incoming>
    <incoming> sequenceID </incoming>
    <outgoing> sequenceID </outgoing>
</exclusiveGateway>
```

Listing 5: XML representation of an XOR-Join gateway with two incoming sequences

```
<sequenceFlow id="sequenceID" sourceRef="predecessorElementID"
targetRef="successorElementID"></sequenceFlow>
```

Listing 6: XML representation of an sequence flow

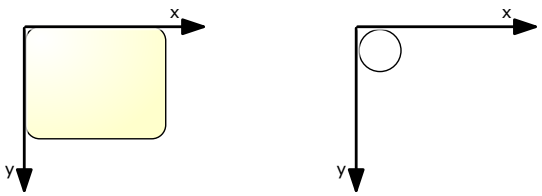

Figure 2.2: Positioning of the elements with x and y axis

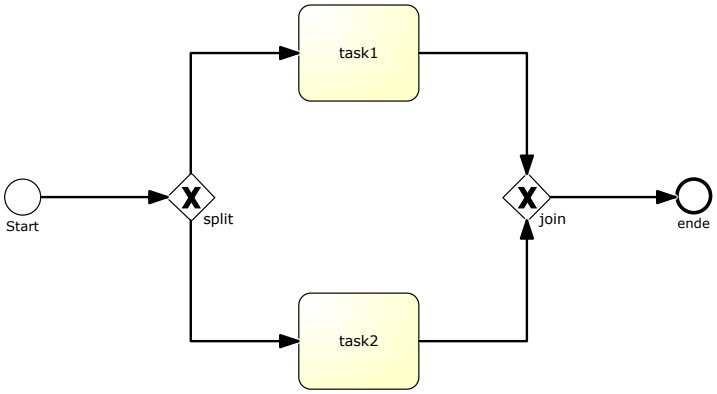

Figure 2.3: A simple BPMN model

at the left upper side, which is also displayed in Figure 2.2. So to display a task with a width of 100 and a height of 80, centered in a square with width 140 and height 120 then the middle of the cell has to be calculated which is:

$$xMiddleOfSquare = 70$$

and

$$yMiddleOfSquare = 60$$

After that the position of the task can be calculated with

$$x = (xMiddleOfSquare) - (halfWidthOfElement)$$

$$y = (yMiddleOfSquare) - (halfHeightOfElement)$$

with the values:

$$x = 70 - (100/2) = 20$$

$$y = 60 - (80/2) = 20$$

These two coordinates are the upper left edge of the task if it should be centered in the square.

For better understanding of the XML of a simple model like in Figure 2.3 the corresponding XML can be found in the Appendix A.1.

3 Psychological Aspects

In this Chapter different psychological aspects are explained. Nudging is explained in Section 3.1. Definitions of different types of motivation and effects on motivation are stated in Section 3.2. The cognitive load theory is described in Section 3.3 and in Section 3.4 color harmonization and color emotion are explained. This is later used for the definition of requirements in Chapter 5. Through the knowledge on those aspects, an application that should be better for learning BPMN can be developed.

3.1 Nudging

In this Section nudging is explained and examples of nudges are given.

First, nudging is the use of nudges to help people make better choices [10]. Even in daily life, nudging is used to direct us, without us noticing it. Nudges are useful because they are choice-preserving approaches to lead people in a specific direction [11]. Normally, the goal of nudges is to create a simpler life for people, whereby it is easier for them to navigate. Nudging works, because people sometimes do not behave rationally [10]. This irrational behavior was also shown in research about cognitive bias [12]. All over the world the interest in nudging increases, because nudges are cheap, deliver fast results, are highly effective and maintain the freedom of choice[11]. A major problem with nudges is, that they should only be used to help people make better choices, but this does not always happen in practice [10]. To understand nudges better, a shortlist of examples is now provided in the following:

- One simple example of nudging is when an architect designs a large building like a university. The architect uses nudging, for example, when he design more open space to create more interaction between people and when he design the stairs at a position where more people can meet he nudges the people towards more possible interaction. The people then can choose if they want to interact or not [13].

- A good example of default rules in nudging is, auto-enrollment into a healthcare plan, because most people tend to accept their enrollment because they understand it is good for them. But if they would not be auto-enrolled they mostly would not make the active choice to enroll [11].
- Another example for nudging would be the simplification of current requirements [11], like the simplification of the tax declaration would lead more people to do their taxes.

More details on nudging and nudges can be found in the work of Sunstein [11], Weinmann et al. [10] and Thaler et al. [13].

3.2 Different Types and Effects on Motivation

This Section provides definitions of different types of motivation and explains effects on motivation.

Firstly, the most widely known types are intrinsic and extrinsic motivation. Intrinsic motivation refers to the performance of an activity with no other reason than performing the process itself [14]. The definition of extrinsic motivation is as follows: The performance of an activity is done because the achieved outcome is valued, instead of the activity itself [14]. An example of intrinsic motivation is a child which plays baseball because it wants to play [15]. The example can be changed a little to be an example for extrinsic motivation: A child plays baseball not only because it wants to but also to please its parents [15]. The downside of extrinsic motivation is, it undermines intrinsic motivation [15]. Because of this, intrinsic motivation is better in the long term. Another theory determines that there are three types of motivation. These are autonomous motivation, controlled motivation, and amotivation [16]. These three types are defined as follows in the work of Gillet et al. [16]:

- Autonomous motivation: This type of motivation refers to performing an activity out of pleasure and own choice.
- Controlled motivation: This type of motivation is defined as performing an activity because of internal (e.g. guilt) or external pressure (e.g. rewards).
- Amotivation: This type refers to the lack of any intention to perform an activity and is relative to the absence of motivation.

All these types can be affected one way or another. For example, they can be influenced by emotions. Emotions influence our natural behavior and the vigor of a response, for example how fast we move [17]. The natural behavior and the vigor of a response influence our motivation. That means emotions influence our motivation not directly but indirectly. In the context of learning, the influence of emotions on motivation was researched in the work of MacIntyre et al. [18]. They showed that positive and negative emotions influence motivation. It is not clear, that negative emotions are "bad" and positive emotions are "good" for motivation and learning because emotions are fundamentally adaptive [18]. Another influence on motivation is shown in the work of Reiss [15], with 16 universal reinforcements. These 16 universal reinforcements influence the motivation of everyone but in different ways. They represent human needs and are Acceptance, Curiosity, Eating, Honor, Idealism, Independence, Order, Physical Activity, Power, Romance, Saving, Social Contact, Status, Tranquility, and Vengeance.

Furthermore, motivation also can trigger emotions through events occurring during the motivation process [17]. Motivation also influences our performance through positive and negative affect, which was shown in Gillet et al. [16].

3.3 Cognitive Load

This Section explains what cognitive load is and from which sources cognitive load can arise.

There is a plethora of research on cognitive load. Cognitive load refers to the working memory, which is needed to carry out the task of learning [19]. Cognitive load is based on cognitive theories, which assume that the human working memory is limited [20]. Because of this limitation of our working memory, cognitive load influences the process of learning. If the cognitive load is reduced, it has a positive impact on the working memory, which provides greater success in learning [21]. Cognitive load can be seen as the construct of representing the working memory resources that are needed for learning or performing a task. The amount of working memory resources that are allocated by the learner for performing the task or the learning is called mental effort [22]. Furthermore, cognitive load can arise from three sources. These three sources are: Intrinsic, Extraneous, and Germane cognitive load [20]. They are explained as follows in the work of Bannert [20]:

- Intrinsic cognitive load: This load is related to the nature of the material that has to be learned. A high intrinsic cognitive load occurs, if the learner does not have enough command over appropriated concepts or it also occurs, if there is a high interactivity between elements [20]. An example for intrinsic cognitive load is, if you have to learn new word pairs of a language, these word pairs have no interaction. Which means that the intrinsic load is low, but if you have to build a sentence, the words have to interact. Through this interaction the intrinsic load will be higher [19].
- Extraneous cognitive load: This kind of load has its origin in a poor way of providing information for a task. It also does not support learning, but only reduces the working memory capacity for learning. An example for high extraneous load is, if the instructions to a task are very difficult to understand, then it is harder to handle the task.
- Germane cognitive load: This load occurs if free working memory is used for the deeper construction and automation of schemata [20]. It is also a bit special, because the other two loads should be held on a low level, but germane load should be increased as far as possible [22].

These three sources of cognitive load influence learning in different ways, for example, reducing the extraneous load and artificially reducing the intrinsic load can help during the learning process [20]. The reduction of extraneous load and increasing of germane load can help to increase the performance of learning too [20]. For a more detailed information on cognitive load, the work of Bannert [20], Schnotz et al. [22] and Kirschner et al. [19] can be useful.

3.4 Color Harmonization and Color Emotions

In this Section color harmonization and color emotion are described. These are valuable concept because emotions influence the motivation and behavior.

It gets more and more interesting for many designers, to know which colors harmonize with each other. There are many concepts about color harmony, one of them by Wilhelm Ostwald. His color circle consisted of 24 hues arranged in a wheel, with a gray core that is black at the bottom and white at the top. The colors were gradated towards the mid in tones. This system looks like a sphere of colors where yellow and blue are on opposite

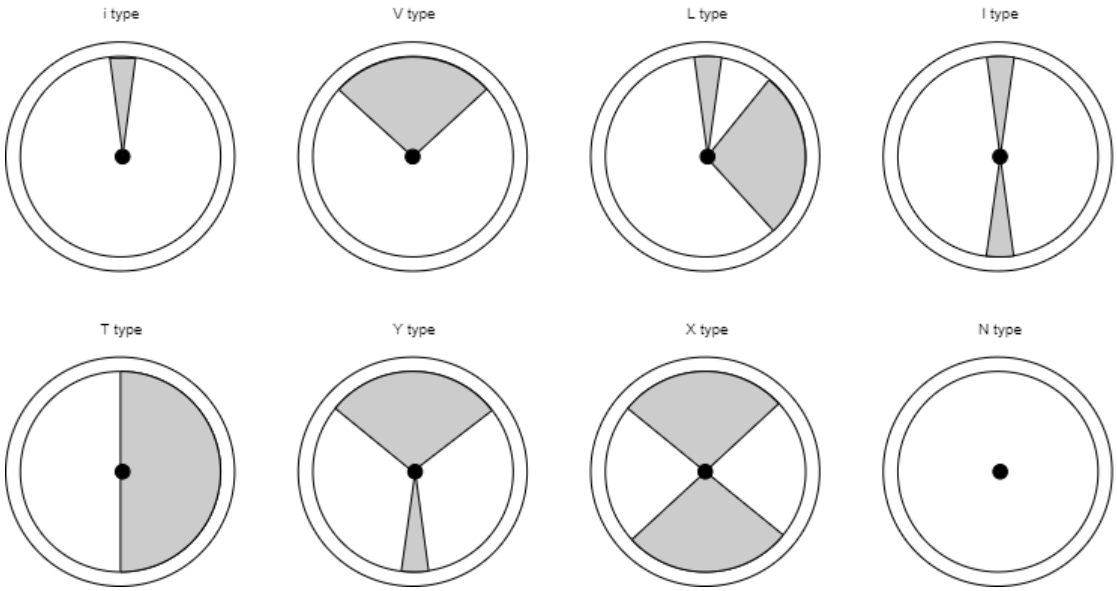

Figure 3.1: Harmonic templates developed by Matsuda. The size of the area can be found in the appendix A.2

sides, as are red and green [23]. The idea of Ostwald is that colors harmonize if they have the same hue or lightness [23]. Another concept of color harmony is from the art instructor and painter Albert Munsell. That system is based on the system of Ostwald but uses value, chroma, and hue [23]. His idea of color harmonization is defined as follows in Westland et al. [23]:

- Colors harmonize if they are of the same hue and chroma
- Colors harmonize if they are complementary and have the same value and chroma
- Colors harmonize if they are in a diminishing sequence, where each color loses one step in value and chroma

One other concept was developed by Matsuda. This concept uses the different harmonic types on a hue channel of the HSV color wheel [24]. These different types can be viewed in Figure 3.1, all colors which fall into the gray area of the templates are defined as harmonic. A problem with all these concepts is, that they are only theories and lack evidence [23]. It is hard to prove these theories because over time and through different aspects like cultural background, the view on colors changes [23].

Not only the harmonization of colors is important to know, but it is also important to know which colors are associated with which emotions. These emotions/feelings which are evoked by colors or a combination of colors are called color emotions [25]. To find associated emotions of colors and their color combination, research was provided from Ou et al. [25, 26]. In their study, they found, that the cultural background influences some of the emotions associated with a color and also found a connection between warm–cool as an emotion ranking and hue, because the red-orange-yellow hue angles are associated with warm, while green-blue-purple hue angles were associated with cool [25]. In their work a sample of 20 colors ranked by color activity, color weight, and color heat can also be found. Color activity can be associated with these emotions as a ranking from fresh to stale, clean to dirty, modern to classical, and active to passive [26]. Furthermore, color weight can be seen as the emotions ranked as tense to relaxed, hard to soft and heavy to light, while color heat is defined by warm to cool [26]. They also showed that changing lightness only influences the emotion ranking of heavy to light and hard to soft [25]. In their second work, they repeated this with pairs of colors, this ranking can be seen in Ou et al. [26]. It is good to know which colors are associated with which emotions, because if you want to create a picture with less emotional weight then you pick colors which have less weight than others, and also harmonize with each other.

4 Related Work

In this Chapter different approaches which are similar to this work will be discussed.

In the last decade, the interest in BPMN and other process modeling languages grew because of the increasing complexity of processes. For this reason much research was conducted. A research that focuses on BPMN modeling tools and their support of guidelines was conducted by Snoeck et al. [27]. The guidelines they used were from Oca et al. [28]. In this work, six modeling tools for BPMN were analyzed for their support on complying on guidelines. The result was, that the majority of creators know the guidelines, but no tool fully supported each guideline. The tool with the highest support level supported only 57.14% of the guidelines, the other tools only supported 50% or less [27]. Better support of modeling guidelines would benefit modeling beginners because the quality of their models would increase. It would also benefit advanced modelers because some guidelines have a positive influence on model understanding and model correctness [27].

Also, there is research on the processing of textual process descriptions with natural language processing (NLP). The field of NLP is known as computational linguistics and tries to solve practical problems in understanding human languages [29]. In their survey, Otter et al. [29] explain, that NLP is the use of statistical and probabilistic computations alongside machine learning. During the recent years machine learning improved through deep learning and neural networks. Thanks to deep learning and neural networks the flexibility, and processing of NLP can be improved further. In the survey of Otter et al. [29], they explain this in detail.

There are several libraries for complex use of NLP, most of them are implemented in Java or Python. Omran et al. [30] provide an overview of four common NLP libraries through their work. They tested all of these four libraries and found that the libraries identified some tokens differently. With further tests they found that spaCy has the most promising accuracy [30]. For the work of this thesis, the spaCy library [31] was used because it

is well documented and offers many possibilities. In the work of van der Aa et al. [32] they tailored NLP techniques to identify activities and their relations from textual descriptions. With these NLP techniques they created declarative process models [32]. In their approach to extract activities they employed the technique from Friedrich et al. [33]. Friedrich et al. [33] used NLP to analyze textual process descriptions; through syntax parsing, semantic analysis, and anaphora resolution they extracted textual description parts and used these to create a BPMN model. On average, 77% of the models were correctly generated [34]. A major problem of creating a BPMN model with NLP is its low tolerance regarding flaws in natural language. These flaws result in significantly lower model quality [35]. Because of such flaws, which are not easy to process with NLP, it is important to have people who also know how to create good process models and these experts can use their knowledge to interpret the textual description correctly and handle such flaws in natural language. It is also important to have experts who can check the quality of automatically generated process models. But before someone can be an expert, he has to learn to create process models and all the fundamentals. Furthermore, the approach on text to model transformation has good first concepts, but these approaches are only in their infancy and require further improvement. The work of Riefer et al. [35] analyzed text to model approaches and found five different approaches. One of them is the approach from Friedrich et al. [33]. The approach of Friedrich was developed in 2011 and is still a state-of-the-art approach [35]. Riefer also explained that there are two methods of using NLP to analyze text. These two are syntactic and semantic analysis [35]. The semantic analysis is used to understand the semantic consistency of words and phrases, while syntactic analysis annotates the words and phrases of a sentence [35].

5 Requirements Definition

For every software project there should be requirements defined, because they lead to software with better quality [36]. Furthermore, requirements are important to define, because they set a frame for the development of the software and define the functionalities. The requirements are used as a measurement for the software, and if these requirements are met, the desired quality will be ensured. Thus, the requirements have to be defined before developing software. They are important in the pre-programming phase, to know what has to be done. They are needed especially in bigger software projects with multiple teams and companies, to ensure everyone knows what has to be done to create the software and keep high-quality standards [37]. But it should also be ensured that requirements should not be defined only to be defined, because no requirements are better than bad requirements [38].

Therefore, in this Chapter we define the requirements for the development of the application of this work. We define the functional and non-functional requirements which will be explained in their respective Sections. Each requirement will have a priority, which declares which requirements are more important than others. The prioritization will be (+) for high priority, (0) for middle priority and (-) for low priority.

5.1 Functional Requirements

In this Section the functional requirements for the software will be defined.

First, functional requirements specify a function that software must be capable of performing [39]. This means that the software should later have the functions that were defined as functional requirements.

The functional requirements for the application can be seen in Table 5.1

Name	Description	Priority
FR-01	The software should be a web application, which is accessible with all common internet browsers in their latest version	0
FR-02	It should be possible to insert a process description text or other process textual descriptions into an editable text field of the application	+
FR-03	It should be possible to highlight text in the text field	+
FR-04	The highlighting should update if the highlighted element in the text is edited	+
FR-05	It should be possible to highlight text as the following BPMN elements: task, start-, end-event, XOR-, OR-, AND-Split, XOR-, OR-, AND-Join. All of these elements should have different highlighting colors, these colors should harmonize with each other and should also have high color-activity but low color-weight	+
FR-06	To highlight the elements in the text field, buttons with the corresponding chosen color for each type of BPMN element should be provided, with which the color of the highlighting should be set	+
FR-07	The elements which are highlighted should be displayed in a list, where they can be used to create a BPMN 2.0 model	+
FR-08	With the highlighted elements it should also be possible to create BPMN 2.0 models with loops	+
FR-09	The application should not allow to position some elements between other elements, like start-events have to be at the first position, end-events need to be at the end	-
FR-10	The application should generate an XML representation for a BPMN 2.0 model, including the layout for the process model	+
FR-11	A download of the created XML representation of the BPMN 2.0 model should be possible	+

Table 5.0 – continued from previous page

Name	Description	Priority
FR-12	Live updates of changes made in the application should be displayed	0
FR-13	A tutorial should be provided, to help to understand how the application works	+
FR-14	It should be possible to delete a created element out of the list and when a element is deleted this way, the respective highlighting in the text field of the element should also be deleted	+
FR-15	The user interface should be implemented using React	0
FR-16	The application should use a Natural Language Processing library to find keywords for splits, starts, ends and tasks. These words should then have another color than the rest of the text to help the user find faster different BPMN element descriptions in the text	+
FR-17	The application should be designed for desktop and mobile devices	-

Table 5.1: functional requirements

5.2 Non-functional Requirements

In this Section one definition of non-functional requirements will be given, but there are many different definitions some of which are discussed in [36, 40].

One simple definition of non-functional requirements states, that they are used to refer to concerns not related to the functionality of the software [36].

The non-functional requirements for the software of this thesis can be found in Table 5.2

Name	Description	Priority
NFR-01	The application should be easy and intuitive to use, that everyone who starts learning process modeling with BPMN 2.0 does not lose their motivation that easily	+
NFR-02	No data should be stored in a database	+
NFR-03	The application should help learners of BPMN 2.0 to easily create a process model because the cognitive load is kept low	+
NFR-04	The application should have a pleasant interface, with harmonizing colors	0
NFR-05	The application should be easily extensible with more BPMN 2.0 elements	0
NFR-06	The code parts of the application which handle the creation and the layout of the BPMN elements should be tested with at least 65% branch coverage	+
NFR-07	The application should not be too reminiscent of other modeling tools	+

Table 5.2: non-functional requirements

6 Conception of the Web Application and Functionality

This Chapter focuses on the conception phase of the web application. The development of the concept for the web application and how the psychological aspects from Chapter 3 influenced it will be described in Section 6.1. Later in Section 6.2 it is explained with a flowchart how the creation of BPMN XML should function and the interaction of components from the application is explained with a sequence diagram.

6.1 Conception of the Web Application

There is a plethora of different tools for creating a BPMN 2.0 model, for example the modeling tools from Signavio [41] or Camunda [42]. The application of this work should be a web application that implements the requirements defined in Chapter 5. Furthermore, it should also help people who start learning BPMN to improve their learning process and should support the modeling and analysis of the process description. This can be accomplished by including the learning of the psychological aspects from Chapter 3. The very first step during the conception was to combine textual process description input of a model with the modeling. It should also be possible to highlight the textual representation of BPMN elements in the process description and these elements can be used to create the model. Through the combination of description text, highlighting, and modeling the intrinsic cognitive load should be reduced because there no longer exist that much interactivity between the textual description and the modeling task. The extraneous cognitive load could be kept low by providing a tutorial that explains how the application works. Thanks to this reduction, the learning of the creation of process models from a textual description should be much easier. Another positive aspect of this combination could be that the motivation level for beginners in modeling should not decrease, because using the application should not require much effort. The positive aspect

of holding the motivation on the same level and keeping low cognitive load means the intrinsic motivation of the user does not decrease. As we know from Chapter 3 intrinsic motivation is the best long term motivation. A further aspect from Chapter 3 is nudging; nudging should be used in the application as follows:

- Due to the combination of textual description and modeling with one tool, the user will possibly use the tool again because it makes their life easier.
- The tool should emphasize keywords for simple BPMN elements like splits, start-, end-events, and tasks. Through this, the user can more easily find element descriptions in the text and create a better process model much faster.
- If a user has to write a process description and create a BPMN model then they can check while using the tool whether their text describes the process correctly and whether the description is understandable.

These were the first aspects that were relevant for the creation of the web application. Now we come to different application concepts, which were iterated until the final concept was decided on. This first idea can be seen as a mock up in Figure 6.1. There it can be seen that it is possible to insert a textual process description, highlight tasks, events, and gateways in the text field. It can also be seen that the different highlighted elements were placed into different lists, with these lists it was possible to connect the elements with each other to later get a BPMN 2.0 XML. One thing which could probably increase the cognitive load of a user is that there are too many connections between the elements and these decrease the clarity of the whole application. It is also possible, that the application would remind users too much of other modeling tools, which also could decrease the motivation. This concept also does not heed the color harmonization and color emotion requested in the requirements. The first concept which heeds them is the final concept which can be seen in Figure 6.3 and is explained later in this Section.

The second idea can be seen in Figure 6.2 and it shows that this concept has a cleaner interface than the first concept. This concept has more focus on the specific BPMN elements, distinguishes between XOR, AND, OR, Splits, and Joins. With this distinction along with different colors, it is easier to see the difference between each BPMN element in the text field. It also makes it easier to focus on the characteristics of splits and joins. These two aspects should help beginners to focus on the process. The concept also does not look like common modeling tools and it does not display any sequences between the different elements. A major difference between the first concept and this concept was

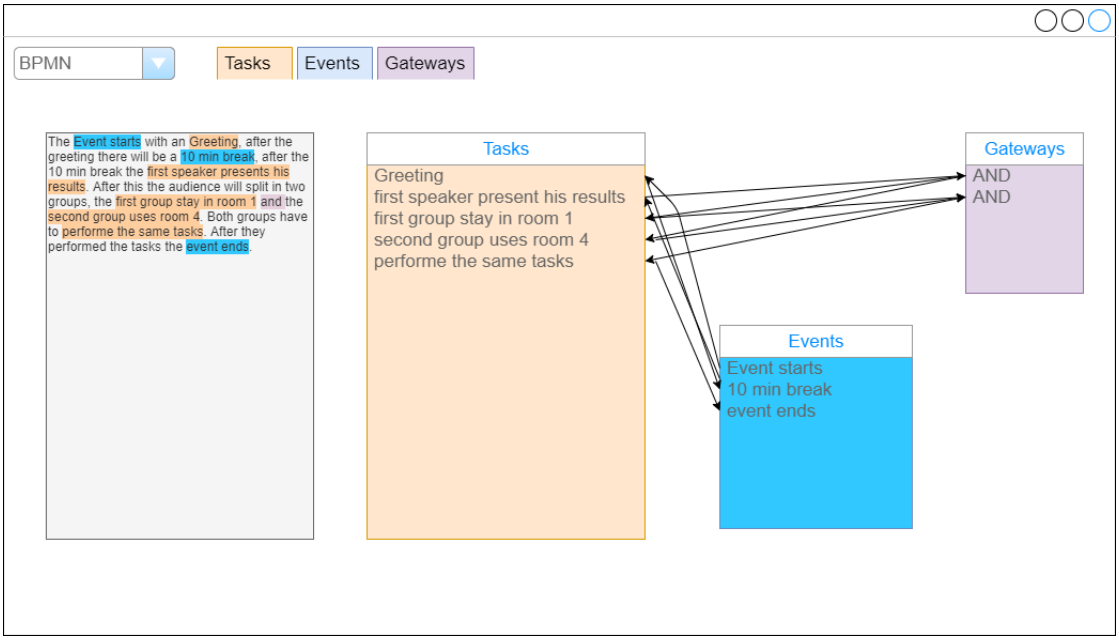

Figure 6.1: First concept of the application

events were not taken into account of modeling a simple BPMN 2.0. Because the start and end events should be created in the background automatically, every other event would make the application more complex, which could lead to decreasing motivation of beginners. Another change of this concept was, that if a split is created the user needs to know how many lists he needs and then the application should create a so-called lane for each list of the split. This would probably reduce the clarity and increase the cognitive load. The major problem was that this concept was not fully sophisticated, because it was not possible to insert joins into lists. It was also not possible to model a loop. One point which is also new in this concept and will be in the further developed concepts is the viewer for the created BPMN 2.0 model. This feature should help to see if the created BPMN is as expected by the user. This should also help users that like common modeling tools and need to see the model they are creating.

The penultimate concept for the application can be seen in Figure 6.3. In this concept it can be seen that it distinguishes between the different elements like in the second concept, while in this concept it is possible to create start and end events explicitly. This is because the application should help the user, but should not limit their decisions where the model starts and ends. Furthermore, as a default every element which is highlighted gets inserted into a list below the text field, except for the joins, which are inserted into

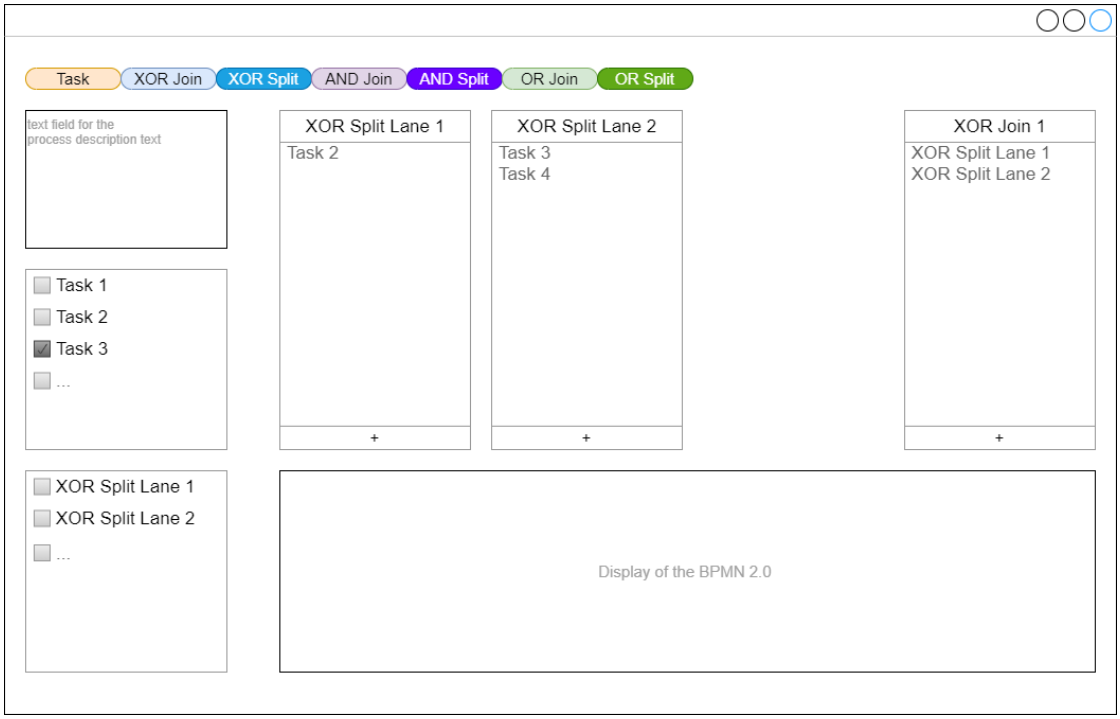

Figure 6.2: Second concept of the application

an separate join list. Another change was that there is by default a list called Main Lane, this list is needed to start modeling. From now on the lists will be called lanes which should not be mistaken with the Lanes of BPMN 2.0. The Main Lane can also have multiple lanes because a model can have multiple start events. The only other elements which have more than one lane are splits. In this concept, the elements are draggable and droppable into the different lanes. It is also possible to click onto a join or split which is placed into a lane. If a join or split is clicked a list representing it opens up. It is then possible to insert elements into their lanes like seen in Figure 6.3 by XOR Split 1 and XOR Join 1. In Figure 6.3 it can also be seen how a loop can be modeled. In this concept it is also possible to directly create joins without highlighting in the textual description, because a join does not always have a representation in a process description. For this, there are three join buttons below the join list. It is also possible to open a tutorial for the usage of the application via the "How to Use?" button. With the Create XML button it is possible to download the XML of the BPMN 2.0 model.

The final concept can be seen in Figure 6.4, this is a refinement of the third concept. The positioning of each list, buttons, and text field were changed to provide a better overview

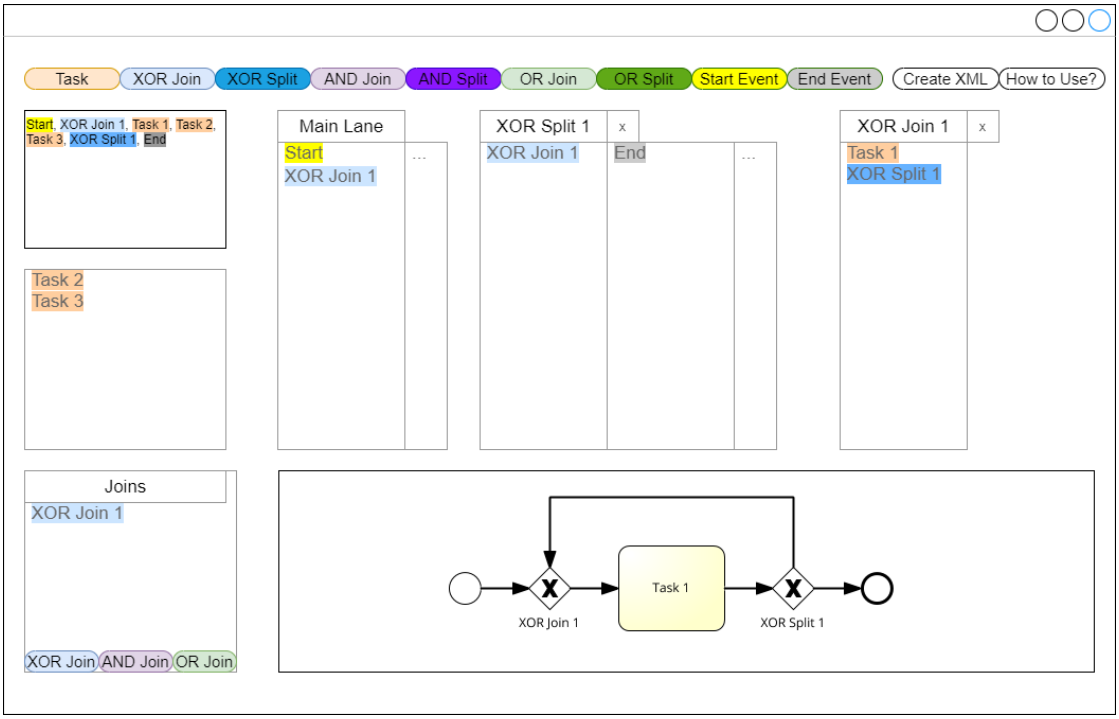

Figure 6.3: Penultimate concept of the application

of each element of the application. Furthermore, in this concept it can be seen that a garbage can symbol is provided, onto which it is possible to drop elements which should be deleted from a lane. A major change was the color of the buttons for the BPMN 2.0 elements, these colors were not chosen randomly. The colors were chosen with help of the knowledge of color emotion and color harmonization. For this four colors were chosen from the work of Ou et al. [25], these colors all had a good color activity, while they also had an average or better color weight. The color heat of the three colors was also very good. These colors were also chosen because they harmonize with each other. The harmonic template T type which is explained in Section 3.4 was used. The hexadecimal value of these four colors are as follows:

- #EBBBBB
- #FFD76F
- #B99AE6
- #EEC209

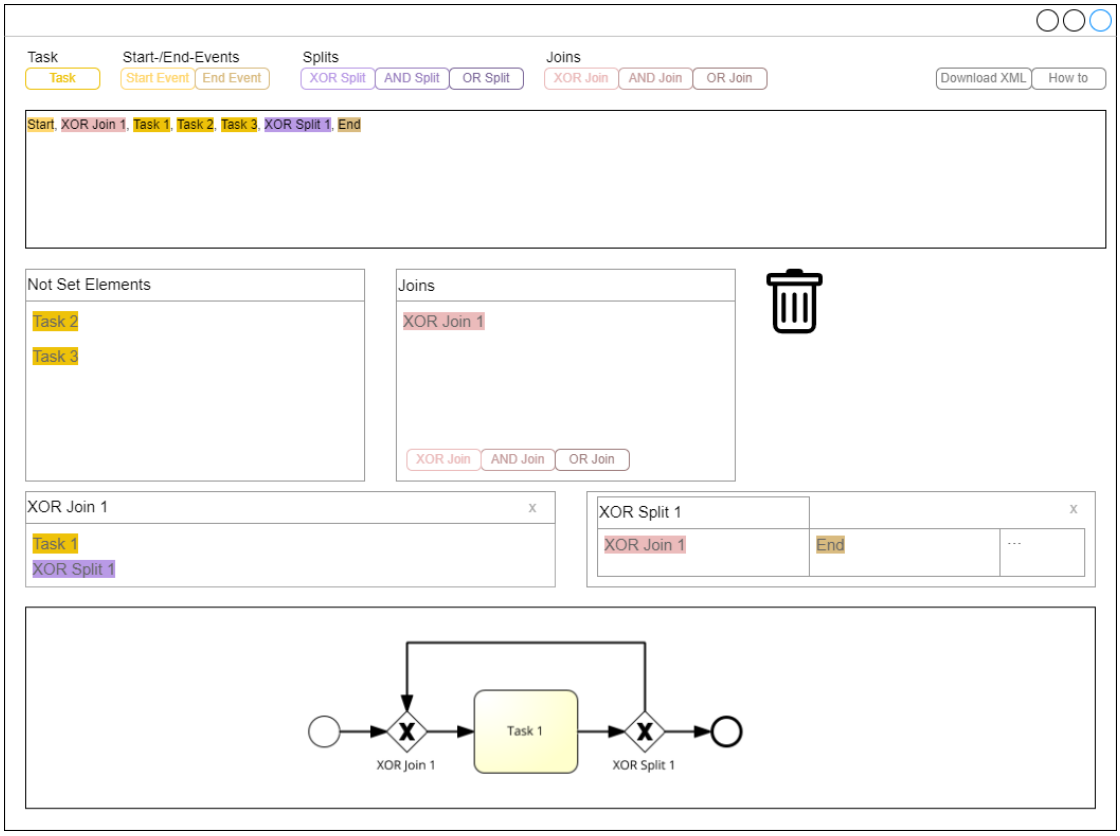

Figure 6.4: Finale concept of the application

A problem occurred because there are more than four buttons. To solve this the buttons of split, join, and start/end were grouped. The four colors were used for the first button of each group and the lightness of the color to the next button in the group was decreased. Through this, the buttons and highlighted text can be distinguished. This decrease of lightness should not have much effect on color weight because the three colors had the best color weight ranking before.

6.2 Conception of the Functionality

In this Section, a flowchart is used to explain the creation of a BPMN XML and a sequence diagram is used to describe the interaction of the user and components during the use of the application.

The creation of a BPMN XML should be possible as follows: Everything begins with the

user starting the application, then inserting the text into the text field and highlighting the BPMN elements in the textual description. After the highlighting of the elements, the elements should be dragged into the correct lane at the correct position. If the correct lane is not open the lane has to be opened first. After placing all elements, the *Download XML* button can be pressed and the BPMN XML should be downloaded. For a better understanding, the whole procedure can be seen in Figure 6.5.

In Figure 6.6 can be seen how the interaction between the user and the application should work. There can be also seen that a component called TagManager should handle the highlighted elements and also manage the lists of the elements. Alongside the TagManager, an XMLGenerator should take the lists from the TagManager and create an XML of the BPMN model. The most active part of the application is the Interface because this accepts all button inputs and displays everything.

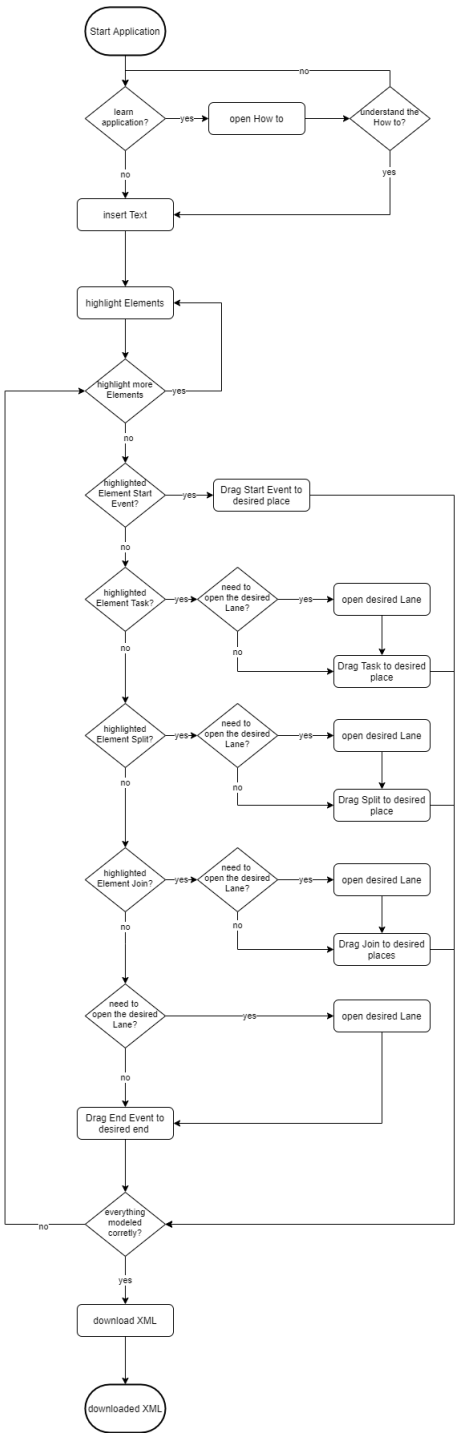

Figure 6.5: Flowchart of creating a BPMN XML with the application

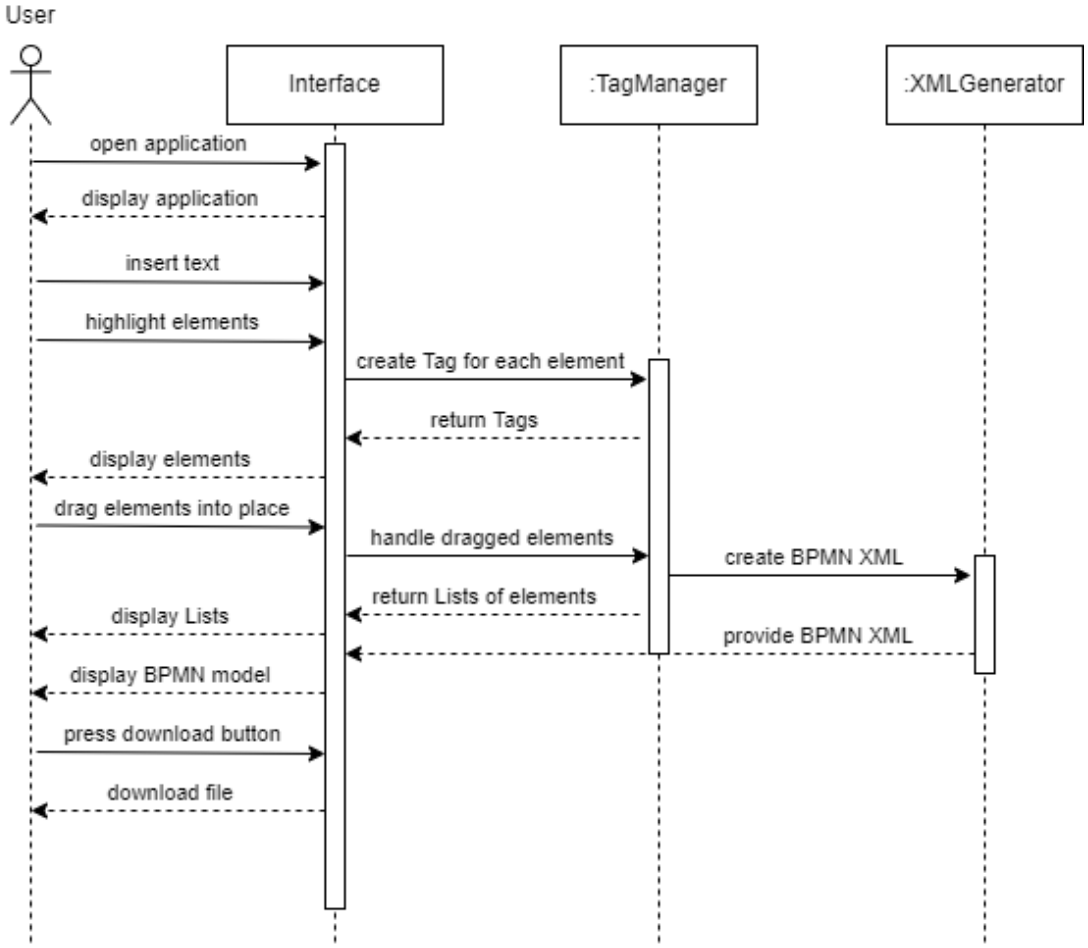

Figure 6.6: The sequence diagram of the usage of the application

7 Development

In this Chapter, the development of the application will be explained through different code examples along with the libraries which were used. It will be explained in Section 7.1 for which purpose the different libraries were needed. In Section 7.2 the architecture of the application will be shown. Later, in Section 7.3 a short explanation of important parts of the implementation will be given. After that, Section 7.4 explains the problems which were encountered, and how they were solved.

7.1 Technology

In this Section the technologies and libraries which were used for the creation of the application are presented and the reason why they were used is given.

7.1.1 Node.js

Of Node.js [43] Version 14.16.1 was used during the development of the application. Node.js is an Open Source JavaScript runtime environment, which is designed to build scalable network applications [43]. In Node.js the node package manager (npm) is included. This package manager provides access to many libraries. The npm handles the installation of libraries and manages them. With npm it is also possible to start scripts. For the creation of the web application, a Node.js development server was used as well as npm for the management and installation of packages and libraries.

7.1.2 React with Typescript

React is a library built for creating user interfaces with JavaScript [44]. It is also possible to use React with TypeScript. React is well documented and there exist many guides and examples on how to use React.

TypeScript [45] is a programming language that builds on JavaScript. It adds type syntax to JavaScript. Through this errors can be found earlier [45]. React was used to create a state-of-the-art web interface with a well-documented framework, while TypeScript was used to find errors through the type syntax. The version of React which was used was v17.0.2, the version of TypeScript was v4.4.3. For quality insurance of the application, tests were created with Jest [46]. Jest comes included within React as a testing library.

7.1.3 Draft.js

Draft.js [47] is a framework for building rich text editors in React. It allows using multiple inline text styles and it is possible to create your own inline text styles [47]. This framework was used to create the text field and the highlighting part of the text field. To create the highlighting, the decorator component of Draft.js was used, this decorator makes it possible to create an HTML span Tag around the text. This span Tag then can be styled with CSS. Version 0.11.7 from Draft.js was used.

7.1.4 React-beautiful-dnd

React-beautiful-dnd [48] is a library that allows the creation of drag and drop interactions within React, with a special focus on lists. This library was used to create a drag-drop environments, where it was possible to insert areas where draggable elements can be dropped. With these possibilities, the lists and draggable elements were created. The library was used in version 13.1.0.

7.1.5 Bpmn-js

Bpmn-js [49] is part of the library bpmn.io, but can also be used separately. Bpmn-js is a BPMN 2.0 rendering toolkit and web modeler [49]. It provides a component to view BPMN 2.0 models and a modeler component, which makes it possible to model BPMN

2.0 models. For the application, only the viewer component was used, to display the created model. For this, version 8.7.3 was used.

7.1.6 Python with Django Framework

For the development of the required Back-End, the Django framework [50] was used in Python [51]. Python is an open source programming language, which has many areas of application, including web development, scientific and numeric computing, and software development [51]. The version of Python which was used is v3.9. Django was used to create an API which made it possible to use the NLP library spaCy and use the results from spaCy in the front-end. Django is a high-level Python web framework, which makes it possible to develop easy and fast web applications [50]. The Django library was used in version 3.2.9.

7.1.7 Natural Language Processing Library SpaCy

The NLP library spaCy [31] is a powerful state-of-the-art NLP python library. SpaCy supports over 64 languages, provides many pretrained functionalities and is easily extensible with custom components and attributes [31]. It was used to find all verbs and direct objects in the text which was inserted into the text field. Through this, it was possible to change the text color of those words, which should help the user find BPMN Task element representations in the text because task elements consist of verb object combinations. Finding the words was possible with the part-of-speech tagging of spaCy. Version 3.2.0 of spaCy was used.

7.1.8 Other Libraries

This Subsection focus on Bootstrap [52] and the file-saver library [53]. Bootstrap is a popular open-source framework for the design and building/customizing of responsive sites [52]. It was used in combination with bootstrap icons for the responsive customizing and styling of the application alongside common CSS. Version 5.1.2 was used of Bootstrap.

The file-saver library makes it easier to generate and save files on the client-side [53]. This functionality was used for the generation of the BPMN XML file, because of this,

the file could be created on the client-side and the download of the file was solved. For this, the 2.0.5 version was used.

7.2 Software Architecture

This Section describes the architecture of the created application.

Since the application is a web application, the web files need to be available on a server. With a web browser, the client requests the web files, then the server sends the files. The web application runs in the web browser and only the NLP is processed in the back-end. This can be seen in Figure 7.1. There can be also seen that the web application consists of the libraries, which interact with the React interface. The two components TagManager and XMLGenerator also interact with the React interface. These two components are a collection of classes and functions, which also use the libraries through the React interface. The TagManager is explained in Subsection 7.3.2, while a part of the XMLGenerator is explained in Subsection 7.3.3. The back-end receives information from the web application, these information are only the text which is inserted into the text field. These information are processed with spaCy, which then provides through a HTTP request the needed information for the NLP component. Through the React interface, input is provided to all the other components, with Bootstrap and CSS the interface gets its styling and can be displayed in the web browser.

7.3 Implementation

In this Section, important parts of the implementation and how the application works because of them will be explained.

First, to implement the text field with highlighting and the drag/drop elements of the interface, the libraries named in Section 7.1 were used. With the help of these libraries, a lot of complexity was outsourced, and it was not needed to program their functionalities. The NLP component which can be seen in Section 7.2 uses the spaCy library. This component will be explained in Subsection 7.3.1. Now a short look on two user interface elements: when the user selects a part of the text and assigns it as a specific BPMN element via one of the buttons then this element is highlighted in the correct color and is added as a draggable element. For this, an interval representing the range of the selected

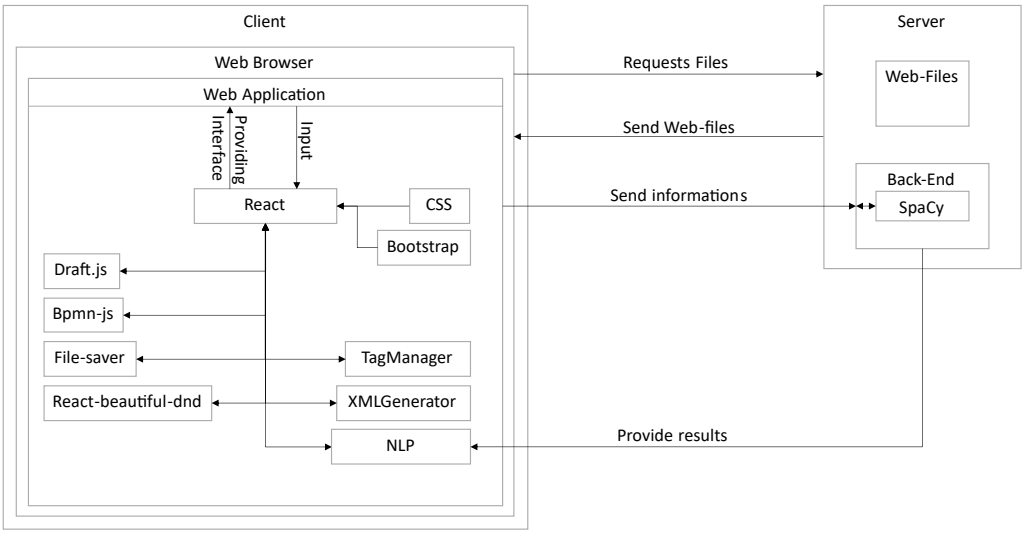

Figure 7.1: The Architecture model of the Web Application

text is created and gets a unique ID. After that, a draggable element will be created for the corresponding text selection. The ID of the element will be called Tag ID, more on this will be explained in Subsection 7.3.2. With the Tag ID, the interval of the draggable element can be looked up and the respective text section can be displayed. Some of these draggable elements are clickable if they are already set into a so-called lane. Lanes are explained in short in Chapter 6. Later in Subsection 7.3.3 it will be explained how the layout of the BPMN XML was solved.

7.3.1 NLP Component

This component uses the spaCy library through the Django API. It passes spaCy the text which was inserted in the text field. Then spaCy response to the NLP component with an array containing every word of the text with its part-of-speech category and its syntactic dependency. Then the NLP component filters every word which has the part-of-speech category 'VERB' or syntactic dependency of a direct object. After the array is filtered, the corresponding interval of the word is searched and is used to set the text color. The NLP component also searches for keywords of BPMN elements like splits, start-, and end-events. The keywords for splits are 'If', 'Or', 'And', 'Else', 'Otherwise', and 'In parallel'. The keywords for start-events are 'Start', 'Starts', and 'Begins'. The

keywords for end-events are 'End', 'Finished', and 'Ends'. The interval for these keywords is searched, and then the text color is changed with the help of this interval.

7.3.2 Tag Components

A central aspect of the implementation are the Tags. These are needed for the whole handling of highlighted text elements which are seen as representation of the corresponding BPMN element. Before the handling of the elements and their intervals will be explained in detail, the structure of tags will be explained.

The structure of Tag is quite simple. The Tag class is defined with multiple abstract methods, which can be seen in Listing 7. Moreover, the Tag class is extended by multiple classes, which represent different BPMN elements that have different behaviors. Because of this, the methods need to be implemented differently. The structure of Tag and the extending classes can be seen in Figure 7.2. The difference in the methods of the extending classes can be seen in Listing 8 and Listing 9; there can be seen a method called *removeParent*. This method is different for JoinTag and NonJoinTag because the Tags representing a join can have more than one parent. While join Tags can have more parents, all other types of BPMN elements (NonJoinTags) can only exist in one lane and have only one parent. Because they have slight differences like these, they need different implementations of these methods.

After this short explanation of Tags, the handling of the highlighted elements can be explained further. For the whole handling of the Tag elements like the deletion, creation of a new Tag and the insertion of Tags into lanes, various functions in the *tagManager* were created. This *tagManager* also provides different lists of Tags, as for example Tags that are not set into a lane or all join Tags. These functions are needed to correctly display the various draggable elements in the interface. But to correctly handle the highlighting in the text field more was needed. For this the *TextToTagManager* class was created. This class maintains the intervals of the highlights, the mapping to their respective ID and helps to provide the correct string of Tags and handles the mapping to create a Tag. The *TextToTagManager* reacts to changes in the text field and checks which intervals need to be updated or deleted. In case of deletion, the respective function of *tagManager* is called. Examples for type of changes to the intervals which the *TextToTagManager* has to manage are the deletion of text in front of the interval seen in Figure 7.3. There can be seen that the interval has to be reduced on both boundaries because of this deletion.

```
export abstract class Tag {
    constructor(){
        newTag(this)
    }
    public id: string = "";

    public abstract toString(): string;

    public abstract getChildren(): string [];
    public abstract removeChild(childID: string): void;
    public abstract addChild(childID: string): void;
    public abstract getParents(): string [];
    public abstract removeParent(parentID: string): void;
    public abstract addParent(parentID: string): void;
}
```

Listing 7: The Tag class which extends JoinTag and NonJoinTag

Another example can be seen in Figure 7.4. There can be seen that if something is inserted in front of the interval, both boundaries need to be increased to represent the correct content. As a third example, in Figure 7.5 can be seen that if something inside the interval is deleted, then the higher boundary needs to be reduced. There are plenty more changes of the text which influence the interval, all this is handled by the *TextToTagManager*.

```
public removeParent(parentID: string){
    const parentIndex = this.parentIDs.indexOf(parentID);
    if (parentIndex >= 0) {
        this.parentIDs.splice(parentIndex,1);
        getTag(parentID)?.removeChild(this.id);
    }
}
```

Listing 8: removeParent(parentID: string) of JoinTag

```
public removeParent(parentID: string){
    if (parentID === this.parentID) {
        this.parentID = "";
        getTag(parentID)?.removeChild(this.id);
    }
}
```

Listing 9: removeParent(parentID: string) of NonJoinTag

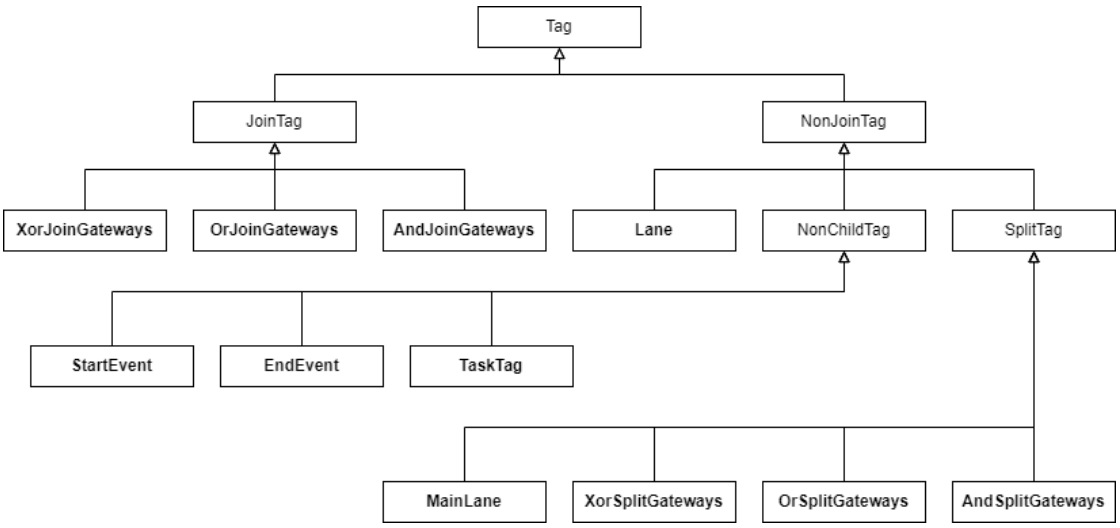

Figure 7.2: Structure of Tag and extending classes, the bold classes can be instantiated

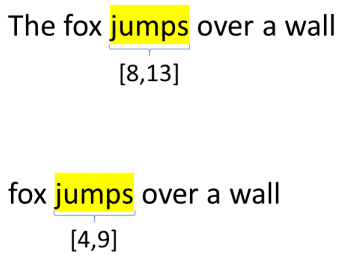

Figure 7.3: The interval changes if something is deleted in front of it

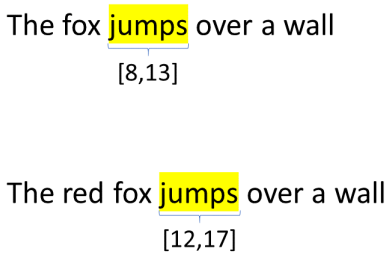

Figure 7.4: The interval changes if something is inserted in front of the interval

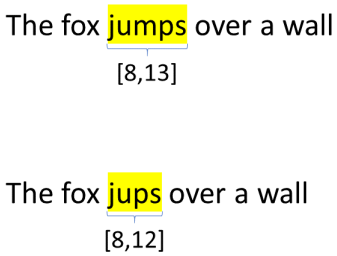

Figure 7.5: The interval changes if something is deleted inside of the interval

7.3.3 Layout of the BPMN

In this Subsection it will be explained how the creation of the layout of the BPMN model was solved.

To layout the BPMN elements an array of triples representing the sequence flows was created. These triples have a predecessor element ID, a successor element ID and a sequence ID. With this array, another array of different types of triples was created, each assigning an element ID its number of incoming sequence flows (called the incoming sequence count) and the number of those sequence flows that were already processed (called the processed sequence count; initialized to be zero). This is needed for later use in the topological sort. The idea for the topological sort was adept from the work of Kitzmann et al. [54]. To get an ordered list, which is needed to correctly set the elements into the grid explained later the topological sort needs to be performed. A version of this sort can be found in the Algorithm 7.3.1. The algorithm takes an element with no unprocessed sequence flows (i.e. processed sequence count = incoming sequence count) or, if no such element can be found, an element with the maximum processed sequence count. If the selected element still has unprocessed incoming sequence flows, these are declared as loop edges and are then removed. Furthermore, the element which was processed is appended to an ordered list, and all outgoing sequence flows of this element are declared as processed sequence flows and the respective triples of the other elements are updated.

The result of the topological sort is a list with sorted elements and a list of loop edges. These two are needed to create the layout because with the list of sorted elements a grid can be created. The grid is made of multiple rows which save the elementIDs and the corresponding column position inside an array. They store the information of which lanes are included in those rows. Furthermore, they need to know the next column position. Through this row structure the elements from the sorted list can be inserted into the grid from the first element to the last element of the list. Each element type needs to be processed differently, for this their Tag representation is used. The simplest Tags are *TaskTag* and *EndEvent*, these two are simply inserted into the same row as their predecessor, where they can be inserted at the next column position. A *StartEvent* Tag always creates a new row in the grid and is inserted into the first column of that row. The two more complex Tag types are *SplitTags*, which create one or more lanes, and *JoinTags*, which create only one lane. *SplitTags* need to be handled as follows: They are inserted into the same row as their predecessor, same as *TaskTag* and *EndEvent*, but

Algorithm 7.3.1 Pseudocode of the topological sort

```
sequenceFlowArray← array of triples with predecessorID, successorID, edgeID;
incomingSeqCounts← array of triples with elementID, number of incoming sequence
flows, number of processed incoming sequence flows (initialized as 0);
orderedList← {}; array to contain the topological sorted element IDs
loopEdgeArray← {}; array to contain all sequence flow IDs considered as loop edges
while incomingSeqCounts not empty do
  triple← an element in incomingSeqCounts with incoming sequence flows = number
  of processed;
  if no such element can be found then
    triple← an element in incomingSeqCounts with highest processed sequence flow
    number;
    loopEdges← all edgeIDs from sequenceFlowArray where successorID = triple el-
    ementID and predecessorID ∉ orderedList;
    for all loopEdges do
      loopEdgeArray← loopEdgeArray ∪ loopEdges;
    end for
  end if
  orderedList← orderedList ∪ {elementID from triple};
  incomingSeqCounts← incomingSeqCounts\{triple};
  newIncomingSeqCount← {};
  for all x in incomingSeqCounts do
    processedEdges ←{e| e ∈ sequenceFlowArray, triple.elementID = e.predecessorID,
    x.elementId = e.successorID };
    newIncomingSeqCount ← newIncomingSeqCount ∪ {(x.elementID, x.incoming,
    x.processed + |processedEdges|)}
  end for
  incomingSeqCounts← newIncomingSeqCount;
end while
return  loopEdgeArray, orderedList;
```

they also create new rows for each outgoing sequence flow, unless they have an uneven number of child lanes, where the middle lane will be inserted into the same row as the *SplitTag*. For the other sequence flows new rows will be created. Every new row inserts the elements to the right of the split. There is also a 'Corner' element inserted at the same column position as the inserted split into each new row which is later needed for the interleaving. *JoinTags* on the other hand need to processed as follows: They calculate their row height by taking the mean of the highest and lowest parent row positions. Then a new row is inserted at this position with a next column position of the highest next column position of the parents. After that, the join is inserted into this new row like any other element. It is necessary to create a new row because otherwise, it is possible that a join intersects a sequence flow between two other elements or gets pushed needlessly far to the right. After all these insertions, every element of the sorted list is inserted into the grid and the whole grid needs to be checked if interleaving is possible. The interleaving is only possible if two rows are adjacent and no elements of both rows overlap (i.e. share column position). Here, the 'Corner' elements prevent unwanted interleaving of splits, because if there would be no 'Corner' elements, Figure 7.6 would have one task right between the split and join. After the interleaving, the 'Corner' elements are deleted.

Now the grid is used to create the layout of the elements. Every element is positioned in the center of the square as explained in Section 2.2. With these positions, the XML can be easily generated for every element and the whole sequence flow. The loop edges are created like every other sequence flow, but they get set on the line between two rows of the grid in order to not overlap other sequence flows. An example of the grid and the positions of the elements can be seen in Figure 7.6.

7.4 Problems during the Implementation

In this Section, the major problems during the implementation will be discussed and their solutions explained.

The biggest problem was the auto layout of the BPMN model, because for the layout, exact positions are needed to set the elements in the correct order at their correct place. First, a possible library for this job was searched. One possibility was the bpmn-auto-layout library [55], but this library had two issues: The first issue was, that the library is not maintained anymore. The second issue was, that it was not possible to layout a process with a loop correctly. This problem appeared because it builds a tree structure to

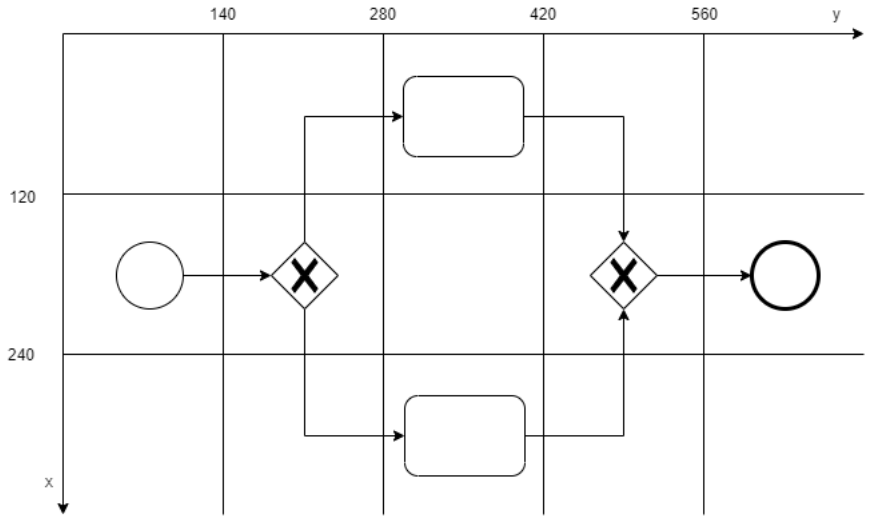

Figure 7.6: An example for the layout grid

position the different elements, a loop caused this tree structure to fail. Because of these two issues, the library was not an option anymore. So, another solution was needed, the best option was provided through the work of Kitzmann et al.[56], which solved the layout through a topological sort. The topological sort was explained in Subsection 7.3.3.

Another problem, which had to be solved, was the highlighting in the text field. This was not a major problem in the beginning, because it was possible to solve the highlighting with help of the library react-highlight-within-textarea [57]. Version 1.0.2 of this library did not cause any problems, not until the CSS and responsive design with bootstrap was implemented. After this the text field with highlighting caused problems when resizing the window. Because of this, the newer version 2.1.0 was tried to use because this should not have any resizing issues. The problem with this version was, that this library would have needed too many changes to use it like intended in the application. Because of this, it was better to use draft-js [47] which was also used by the react-highlight-within-textarea library. With draft-js, the highlighting in the text field was simpler to solve and caused no issue with the resizing of the window.

8 Demonstration of the Web Application

This Chapter presents the final state of the application and how it works.

First, the web application when opened looks like in Figure 8.1. There a button can be seen for each type of BPMN element which was requested in the Requirements Chapter 5. These buttons are used to highlight text sections in the text field; it is possible to copy or insert text into the text field. Furthermore, the highlighted elements can either be seen in the *Not Set Elements* list or the *Joins* list. These elements can be dragged and dropped into other lists, through which it is possible to create BPMN models. A button is provided in the upper right hand side to download the BPMN as an XML, which later can be imported into other modeling tools. More details about the functionality of the application is available in the How-to of the application, which can be found in Appendix A.4 or in the upper right hand side of the application when clicked on the *How To* button.

Now we will have a look at a use example of the application. A textual process description is inserted into the text field; the process description text can be found in the Appendix A.3. After the text is inserted, the application changes the color of some words in the text. Some of these words are keywords for splits, start-, and end-events and others are verbs and direct objects, which are suggestions for tasks. The color of the suggestions is similar to the button color of the elements, this can be seen in Figure 8.2. Now the user can highlight the textual descriptions of BPMN elements, which can be seen in Figure 8.3. After the elements are highlighted, they are displayed in their corresponding list, like displayed in Figure 8.4. The insertion of the elements into the correct list is simple, it always starts with the *Main Lane*. In this example the *Main Lane* has two lists, one for each start-event. No elements can be placed after a split, join or end event in a list, as seen in Figure 8.5. Then the list of the corresponding split/join needs to be opened by clicking on the element and the elements which should be placed after the split/join are inserted into the list of the split/join. This can be seen in Figure 8.6. The elements after the first join are placed into its list. It can also be seen that the *"start*

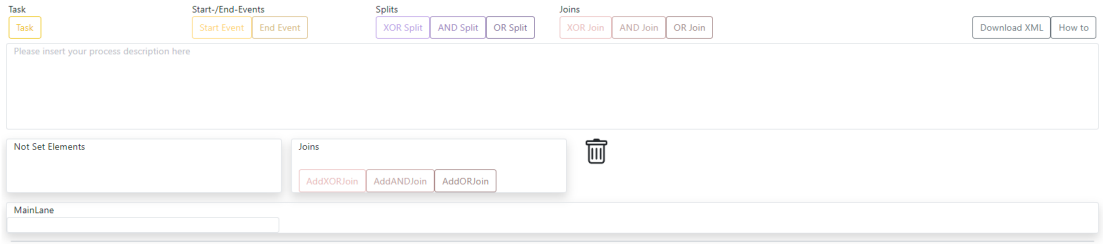

Figure 8.1: The application after the start

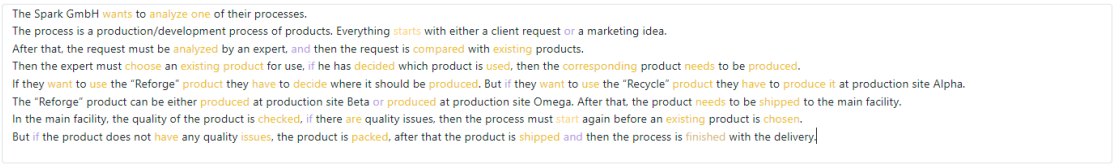

Figure 8.2: The text inserted into the text field with suggestion help

again" join is at the end of this join, because it joins together the list of *"XORJoin1"* and one list of *"quality issues"*. The BPMN model created in this example can be seen in Figure 8.7. The whole creation of the model with the application can be seen in Figure 8.8 as a graph, for better understanding of how the elements need to be inserted into the lists.

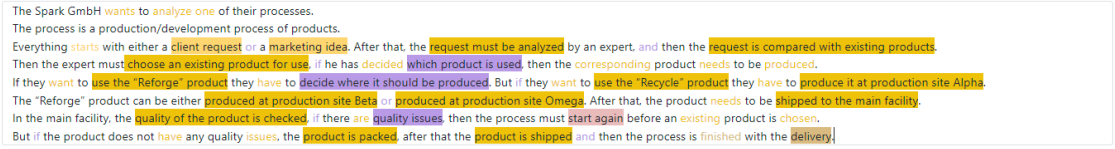

Figure 8.3: The elements highlighted in the text

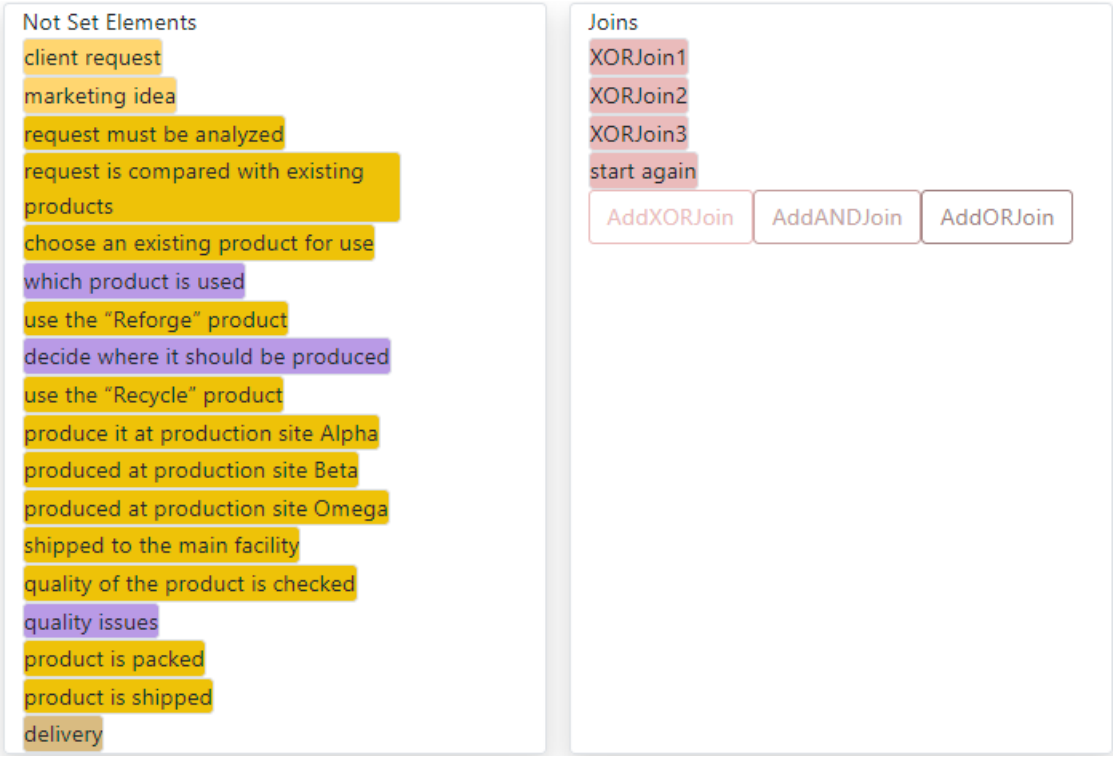

Figure 8.4: The elements if they were highlighted and not set into an existing lane

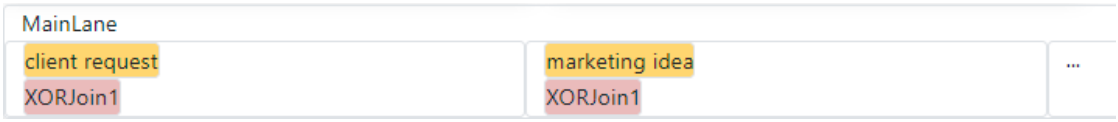

Figure 8.5: The Main Lane with the input of the two start events and the XOR-Join

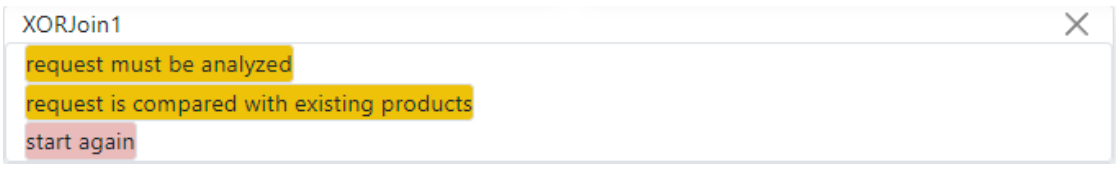

Figure 8.6: The elements after the XOR-Join of the start events

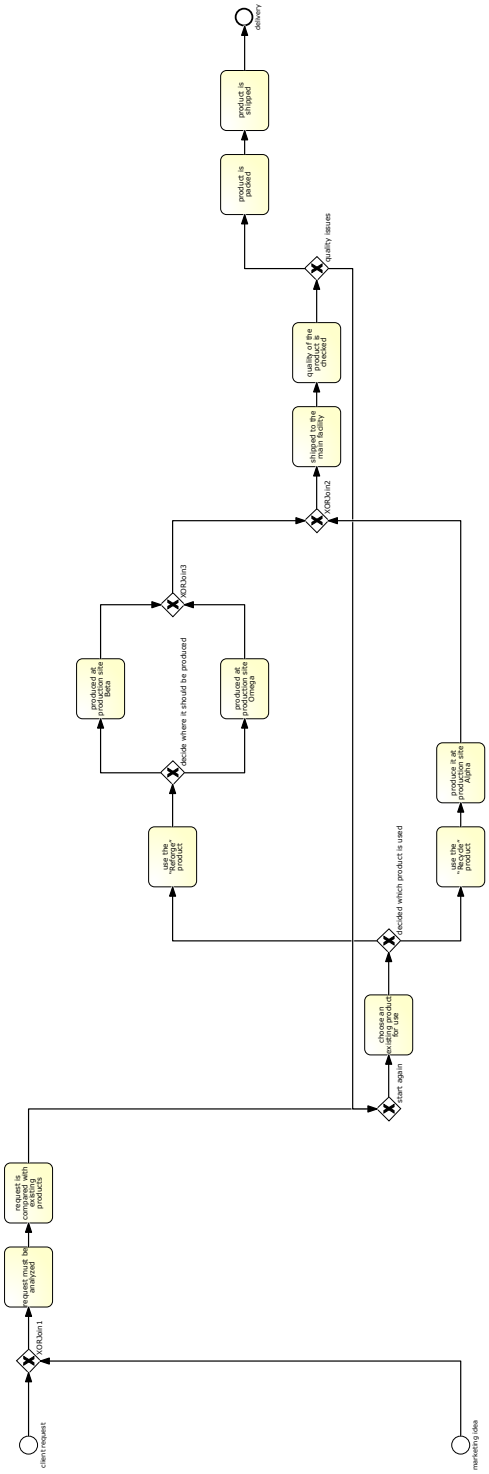

Figure 8.7: The model which was created with the application

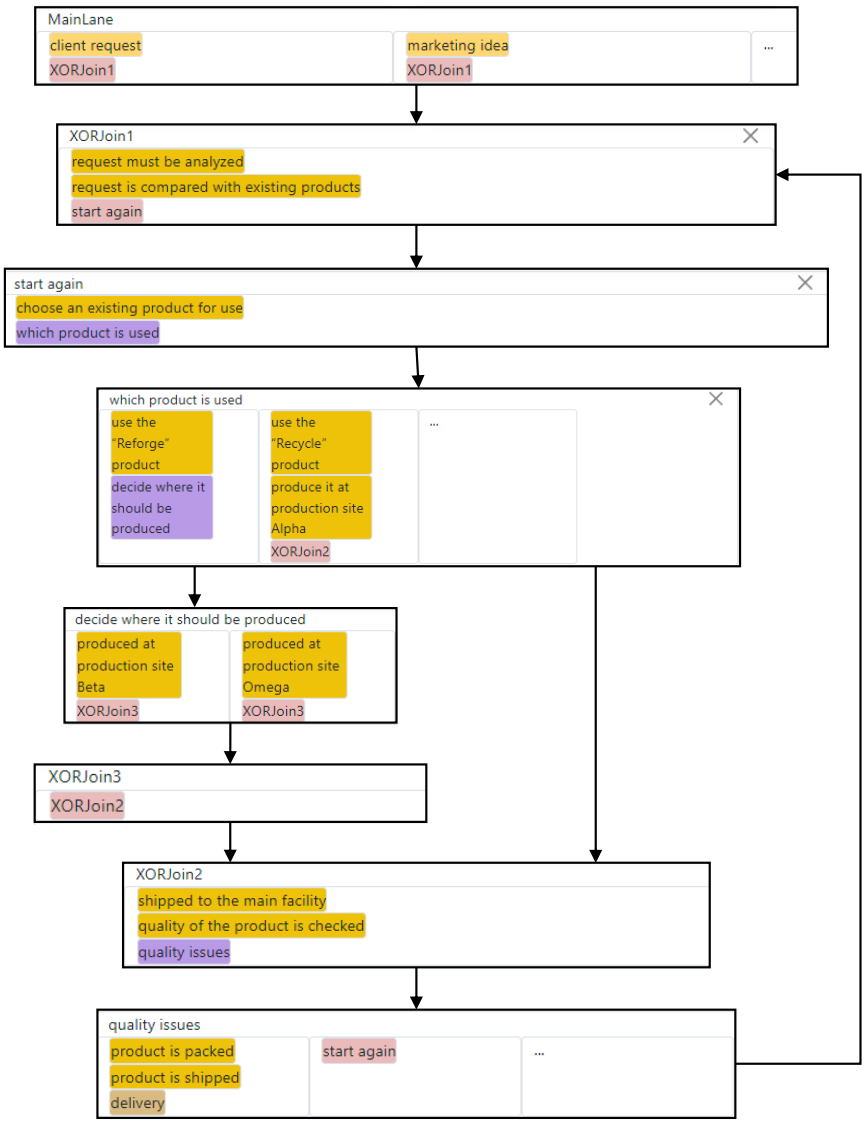

Figure 8.8: All lists for the generation of the process with the application

9 Requirements Checking

In this Chapter the requirements that were defined in Chapter 5 will be checked, on whether they have been met(+), need to be checked [with for example a study] (0) or have not been met (-).

9.1 Checking of the Functional Requirements

In this Section it will be checked, if the application meets the functional requirements. This can be seen in Table 9.1.

Name	Description	Priority	+/0/-
FR-01	The software should be a web application, which is accessible with all common internet browsers in their latest version	0	0
FR-02	It should be possible to insert a process description text or other process textual descriptions into an editable text field of the application	+	+
FR-03	It should be possible to highlight text in the text field	+	+
FR-04	The highlighting should update if the highlighted element in the text is edited	+	+

Table 9.0 – continued from previous page

Name	Description	Priority	+/0/-
FR-05	It should be possible to highlight text as the following BPMN elements: task, start-, end-event, XOR-, OR-, AND-Split, XOR-, OR-, AND-Join. All of these elements should have different highlighting colors, these colors should harmonize with each other and should also have high color-activity but low color-weight	+	+
FR-06	To highlight the elements in the text field, buttons with the corresponding chosen color for each type of BPMN element should be provided, with which the color of the highlighting should be set	+	+
FR-07	The elements which are highlighted should be displayed in a list, where they can be used to create a BPMN 2.0 model	+	+
FR-08	With the highlighted elements it should also be possible to create BPMN 2.0 models with loops	+	+
FR-09	The application should not allow to position some elements between other elements, like start-events have to be at the first position, end-events need to be at the end	-	-
FR-10	The application should generate an XML representation for a BPMN 2.0 model, including the layout for the process model	+	+
FR-11	A download of the created XML representation of the BPMN 2.0 model should be possible	+	+
FR-12	Live updates of changes made in the application should be displayed	0	+
FR-13	A tutorial should be provided, to help to understand how the application works	+	+

Table 9.0 – continued from previous page

Name	Description	Priority	+/0/-
FR-14	It should be possible to delete a created element out of the list and when a element is deleted this way, the respective highlighting in the text field of the element should also be deleted	+	+
FR-15	The user interface should be implemented using React	0	+
FR-16	The application should use a Natural Language Processing Library to find keywords for splits, starts, ends and tasks. These words should then have another color than the rest of the text to help the user find faster different BPMN element descriptions in the text	+	+
FR-17	The application should be designed for desktop and mobile devices	-	-

Table 9.1: functional requirements checking

9.2 Checking of the Non-functional Requirements

In this Section it will be checked, whether the application meets the non-functional requirements. This can be seen in Table 9.2.

Name	Description	Priority	+/0/-
NFR-01	The application should be easy and intuitive to use, that everyone who starts learning process modeling with BPMN 2.0 does not lose their motivation that easily	+	0

Table 9.1 – continued from previous page

Name	Description	Priority	+/0/-
NFR-02	No data should be stored in a database	+	+
NFR-03	The application should help learners of BPMN 2.0 to easily create a process model because the cognitive load is kept low	+	0
NFR-04	The application should have a pleasant interface, with harmonizing colors	0	+
NFR-05	The application should be easily extensible with more BPMN 2.0 elements	0	+
NFR-06	The code parts of the application which handle the creation and the layout of the BPMN elements should be tested with at least 65% branch coverage	+	+
NFR-07	The application should not be too reminiscent of other modeling tools	+	+

Table 9.2: non-functional requirements checking

10 Conclusion and Future Work

In this Chapter, the thesis is summarized and an outlook on future work is provided.

10.1 Conclusion

The objective of this thesis was to create a web application for modeling BPMN models which helps with learning BPMN and can also be used by experts to create simple BPMN models through the use of textual descriptions. In order to achieve the creation of such a web application, knowledge from multiple research areas was used.

First, psychological knowledge on learning and the influences on learning was used. Other information like nudging, associations of colors with emotions and the harmonization of colors were used as well. With this information and basic knowledge of the creation of BPMN XML concepts for the web application were developed, but not before requirements of the application were defined, which were taken into account during the concept phase. During the conception phase, the design of the interface was created. It was also considered how the different parts of the application should interact with each other during the creation of a BPMN XML file. After the concept was created, the application was implemented using React with TypeScript to create the web interface. For the creation of some interface elements, different libraries were used. The application provides a text field, where a textual process description can be inserted. In this text field keywords for different BPMN elements are emphasized through different text colors. To find task elements more easily, a NLP library was used to change the text color of verbs and direct objects. This emphasizing of possible BPMN elements should help the user to find BPMN elements faster. The user can highlight different elements through a corresponding button; these elements are displayed as drag and drop elements in a list and can be used to create a BPMN model. After every element is positioned and the correct BPMN model is displayed in the viewer the corresponding BPMN XML can be downloaded. A demonstration of the application was also shown in this work.

10.2 Future Work

Since this work used the knowledge of psychological aspects during the conception phase, it would be good to confirm whether the application influences the learning process like planned. This could be shown with an eye-tracking study, with the task of modeling BPMN models through textual descriptions. There would be two groups: One of which uses the application and the other uses a normal modeling tool. Then the data can be checked for differences between the two groups. With such a study, the influence of the application on motivation and cognitive load could be verified. Another interesting study could be on the usability of the application. With such a study it would be possible to find weaknesses and improve the application further. Furthermore, it would be possible to incorporate more natural language processing or further improve the keywords used in the NLP component. It could be possible to increase the accuracy of the predicted words with NLP, which should help the user find the BPMN elements in the description. The NLP algorithm could not only be used to find verbs and their direct objects but also to analyze sentences semantically. It could also be possible to use NLP to create a process model from the textual description, which can be used as a guide. There are many possibilities for what could be done in future work, especially with NLP.

Bibliography

[1] R. Petrusel, J. Mendling, and H. A. Reijers, "How visual cognition influences process model comprehension," *Decision Support Systems*, 2017.

[2] M. Zimoch, R. Pryss, J. Schobel, and M. Reichert, "Eye tracking experiments on process model comprehension: Lessons learned," in *Enterprise, Business-Process and Information Systems Modeling*, Springer Berlin Heidelberg, 2017.

[3] M. Indulska, P. Green, J. Recker, and M. Rosemann, "Business process modeling: Perceived benefits," in *Conceptual Modeling - ER 2009*, Springer Berlin Heidelberg, 2009.

[4] M. Chinosi and A. Trombetta, "Bpmn: An introduction to the standard," *Computer Standards & Interfaces*, 2012.

[5] H. Mili, G. Tremblay, G. B. Jaoude, E. Lefebvre, L. Elabed, and G. E. Boussaidi, "Business process modeling languages: Sorting through the alphabet soup," *ACM Comput. Surv.*, 2010.

[6] Object Managemet Group, "About the business process model and notation specification version 2.0," 2021. Last accessed 05 October 2021 `https://www.omg.org/spec/BPMN/2.0/About-BPMN/`.

[7] J. Recker, "Opportunities and constraints: the current struggle with bpmn," *Business Process Management Journal*, 2010.

[8] Object Managemet Group, "About the business process model and notation specification version 2.0.2," 2021. Last accessed 08 October 2021 `https://www.omg.org/spec/BPMN/`.

[9] M. Halaška and R. Šperka, "Importance of process flow and logic criteria for rpa implementation," in *Agents and Multi-Agent Systems: Technologies and Applications 2020*, Springer Singapore, 2020.

[10] M. Weinmann, C. Schneider, and J. vom Brocke, "Digital nudging," *Business & Information Systems Engineering*, 2016.

[11] C. R. Sunstein, "Nudging: a very short guide," *Business Economics*, 2019.

[12] F. Blanco, "Cognitive bias," in *Encyclopedia of Animal Cognition and Behavior*, Springer International Publishing, 2017.

[13] R. H. Thaler and C. R. Sunstein, *Nudge: Wie man kluge Entscheidungen anstößt.* Ullstein Buchverlage, 2008.

[14] T. S. Teo, V. K. Lim, and R. Y. Lai, "Intrinsic and extrinsic motivation in internet usage," *Omega*, 1999.

[15] S. Reiss, "Intrinsic and extrinsic motivation," *Teaching of Psychology*, 2012.

[16] N. Gillet, R. J. Vallerand, M.-A. K. Lafrenière, and J. S. Bureau, "The mediating role of positive and negative affect in the situational motivation-performance relationship," *Motivation and Emotion*, 2013.

[17] M. Popa and I. I. Salanta, "The emotions'role in the motivation process," *Managerial Challenges of the Contemporary Society*, 2013.

[18] P. D. MacIntyre and L. Vincze, "Positive and negative emotions underlie motivation for L2 learning," *Studies in Second Language Learning and Teaching*, 2017.

[19] P. A. Kirschner, J. Sweller, F. Kirschner, and J. Zambrano R., "From cognitive load theory to collaborative cognitive load theory," *International Journal of Computer-Supported Collaborative Learning*, 2018.

[20] M. Bannert, "Managing cognitive load—recent trends in cognitive load theory," *Learning and Instruction*, 2002.

[21] M. Zimoch, R. Pryss, T. Probst, W. Schlee, and M. Reichert, "The repercussions of business process modeling notations on mental load and mental effort," in *Business Process Management Workshops*, Springer International Publishing, 2019.

[22] W. Schnotz and C. Kürschner, "A reconsideration of cognitive load theory," *Educational Psychology Review*, 2007.

[23] S. Westland, K. Laycock, V. Cheung, P. Henry, and F. Mahyar, "Colour harmony," *Color: Design & Creativity*, 2007.

[24] D. Cohen-Or, O. Sorkine, R. Gal, T. Leyvand, and Y.-Q. Xu, "Color harmonization," in *ACM SIGGRAPH 2006 Papers*, Association for Computing Machinery, 2006.

[25] L.-C. Ou, M. R. Luo, A. Woodcock, and A. Wright, "A study of colour emotion and colour preference. part I: Colour emotions for single colours," *Color Research & Application*, 2004.

[26] L.-C. Ou, M. R. Luo, A. Woodcock, and A. Wright, "A study of colour emotion and colour preference. part II: Colour emotions for two-colour combinations," *Color Research & Application*, 2004.

[27] M. Snoeck, I. Moreno-Montes de Oca, T. Haegemans, B. Scheldeman, and T. Hoste, "Testing a selection of bpmn tools for their support of modelling guidelines," in *The Practice of Enterprise Modeling*, Springer International Publishing, 2015.

[28] I. Moreno-Montes de Oca and M. Snoeck, "Pragmatic guidelines for business process modeling," tech. rep., KU Leuven - Faculty of Economics & Business, 2014.

[29] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[30] F. N. A. Al Omran and C. Treude, "Choosing an nlp library for analyzing software documentation: a systematic literature review and a series of experiments," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, 2017.

[31] ExplosionAI GmbH, "spaCy." Last accessed 30 November 2021 `https://spacy.io/`.

[32] H. van der Aa, C. Di Ciccio, H. Leopold, and H. A. Reijers, "Extracting declarative process models from natural language," in *Advanced Information Systems Engineering*, Springer International Publishing, 2019.

[33] F. Friedrich, J. Mendling, and F. Puhlmann, "Process model generation from natural language text," in *Advanced Information Systems Engineering*, Springer Berlin Heidelberg, 2011.

[34] J. Mendling, H. Leopold, L. H. Thom, and H. van der Aa, "Natural language processing with process models (nlp4re report paper)," in *REFSQ Workshops*, 2019.

[35] M. Riefer, S. F. Ternis, and T. Thaler, "Mining process models from natural language text: A state-of-the-art analysis," *Multikonferenz Wirtschaftsinformatik*, 2016.

[36] L. Chung, P. Leite, and J. S. Cesar, "On non-functional requirements in software engineering," in *Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos*, Springer Berlin Heidelberg, 2009.

[37] M. Broy, E. Geisberger, J. Kazmeier, A. Rudorfer, and K. Beetz, "Ein requirements-engineering-referenzmodell," *Informatik-Spektrum*, 2007.

[38] C. Rupp, M. Simon, and F. Hocker, "Requirements engineering und management," *HMD Praxis der Wirtschaftsinformatik*, 2009.

[39] J. W. Brackett, "Software requirements," tech. rep., Carnegie-Mellon Univ. Pittsburgh PA Software Engineering Institute, 1990.

[40] M. Glinz, "On non-functional requirements," in *15th IEEE International Requirements Engineering Conference*, 2007.

[41] Signavio GmbH, "Signavio SAP process manager," 2021. Last accessed 04 October 2021 `https://www.signavio.com/de/products/process-manager/`.

[42] Camunda Services GmbH, "Camunda modeler," 2021. Last accessed 04 October 2021 `https://camunda.com/de/download/modeler/`.

[43] Joyent Inc., "Node.js." Last accessed 18 October 2021 `https://nodejs.org/en/about/`.

[44] Facebook Inc., "React." Last accessed 18 October 2021 `https://reactjs.org/`.

[45] Microsoft, "Typescript." Last accessed 18 October 2021 `https://www.typescriptlang.org/`.

[46] Facebook Inc., "Jest." Last accessed 18 October 2021 `https://jestjs.io/`.

[47] Facebook Inc., "Draft.js." Last accessed 18 October 2021 `https://draftjs.org/`.

[48] Atlassian Pty Ltd, "react-beautiful-dnd." Last accessed 18 October 2021 `https://github.com/atlassian/react-beautiful-dnd`.

[49] Camunda Services GmbH, "bpmn-js." Last accessed 18 October 2021 `https://bpmn.io/toolkit/bpmn-js/`.

[50] Django Software Foundation, "Django." Last accessed 1 December 2021 `https://www.djangoproject.com/`.

[51] Python Software Foundation, "Python." Last accessed 1 December 2021 `https://www.python.org/`.

[52] Twitter, Inc. and The Bootstrap Authors, "Bootstrap." Last accessed 18 October 2021 `https://getbootstrap.com/`.

[53] E. Grey, "Filesaver.js." Last accessed 18 October 2021 `https://github.com/eligrey/FileSaver.js`.

[54] I. Kitzmann, C. König, D. Lübke, and L. Singer, "A simple algorithm for automatic layout of bpmn processes," in *2009 IEEE Conference on Commerce and Enterprise Computing*, 2009.

[55] Camunda Services GmbH, "bpmn-auto-layout." Last accessed 21 October 2021 `https://github.com/bpmn-io/bpmn-auto-layout`.

[56] I. Kitzmann, C. König, D. Lübke, and L. Singer, "A simple algorithm for automatic layout of bpmn processes," in *2009 IEEE Conference on Commerce and Enterprise Computing*, 2009.

[57] M. Eklund, "react-highlight-within-textarea." Last accessed 21 October 2021 `https://github.com/bonafideduck/react-highlight-within-textarea`.

List of Figures

List of Tables

List of Listings

List of Algorithms

A Appendix

A.1 XML of a Simple BPMN model from Figure 2.3

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
     xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
     xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
     xmlns:omgdc="http://www.omg.org/spec/DD/20100524/DC"
     xmlns:omgdi="http://www.omg.org/spec/DD/20100524/DI"
     expressionLanguage="http://www.w3.org/TR/XPath"
     typeLanguage="http://www.w3.org/2001/XMLSchema"
     >
     <process id="ID0" name="GeneratedProcessModel">
          <startEvent id="ID2" name="Start ">
               <outgoing>ID2ID3</outgoing>
          </startEvent>
          <exclusiveGateway gatewayDirection="Diverging" id="ID3"
          ↪  name="split ">
               <incoming>ID2ID3</incoming>
               <outgoing>ID3ID5</outgoing>
               <outgoing>ID3ID6</outgoing>
          </exclusiveGateway>
          <task id="ID5" name="task1 ">
               <incoming>ID3ID5</incoming>
               <outgoing>ID5ID7</outgoing>
          </task>
          <task id="ID6" name="task2 ">
               <incoming>ID3ID6</incoming>
```

```
            <outgoing>ID6ID7</outgoing>
        </task>
        <exclusiveGateway gatewayDirection="Converging" id="ID7"
        ↪  name="join ">
            <incoming>ID5ID7</incoming>
            <incoming>ID6ID7</incoming>
            <outgoing>ID7ID9</outgoing>
        </exclusiveGateway>
        <endEvent id="ID9" name="ende">
            <incoming>ID7ID9</incoming>
        </endEvent>
        <sequenceFlow id="ID2ID3" sourceRef="ID2"
        ↪  targetRef="ID3"></sequenceFlow>
        <sequenceFlow id="ID3ID5" sourceRef="ID3"
        ↪  targetRef="ID5"></sequenceFlow>
        <sequenceFlow id="ID5ID7" sourceRef="ID5"
        ↪  targetRef="ID7"></sequenceFlow>
        <sequenceFlow id="ID7ID9" sourceRef="ID7"
        ↪  targetRef="ID9"></sequenceFlow>
        <sequenceFlow id="ID3ID6" sourceRef="ID3"
        ↪  targetRef="ID6"></sequenceFlow>
        <sequenceFlow id="ID6ID7" sourceRef="ID6"
        ↪  targetRef="ID7"></sequenceFlow>
    </process>
    <bpmndi:BPMNDiagram>
        <bpmndi:BPMNPlane bpmnElement="ID0">
            <bpmndi:BPMNShape bpmnElement="ID5" id="ID5_gui">
                <omgdc:Bounds height="80" width="100" x="300"
                ↪  y="20"/>
            </bpmndi:BPMNShape>
            <bpmndi:BPMNShape bpmnElement="ID2" id="ID2_gui">
                <omgdc:Bounds height="30" width="30" x="55" y="165"/>
            </bpmndi:BPMNShape>
            <bpmndi:BPMNShape bpmnElement="ID3" id="ID3_gui"
            ↪  isMarkerVisible="true">
```

```
        <omgdc:Bounds height="30" width="30" x="195"
        ↪  y="165"/>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape bpmnElement="ID7" id="ID7_gui"
    ↪  isMarkerVisible="true">
        <omgdc:Bounds height="30" width="30" x="475"
        ↪  y="165"/>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape bpmnElement="ID9" id="ID9_gui">
        <omgdc:Bounds height="30" width="30" x="615"
        ↪  y="165"/>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape bpmnElement="ID6" id="ID6_gui">
        <omgdc:Bounds height="80" width="100" x="300"
        ↪  y="260"/>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNEdge bpmnElement="ID2ID3" id="ID2ID3_gui">
        <omgdi:waypoint x="85" y="180"/>
        <omgdi:waypoint x="195" y="180"/>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge bpmnElement="ID3ID5" id="ID3ID5_gui">
        <omgdi:waypoint x="210" y="195"/>
        <omgdi:waypoint x="210" y="60"/>
        <omgdi:waypoint x="300" y="60"/>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge bpmnElement="ID5ID7" id="ID5ID7_gui">
        <omgdi:waypoint x="400" y="60"/>
        <omgdi:waypoint x="490" y="60"/>
        <omgdi:waypoint x="490" y="165"/>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge bpmnElement="ID7ID9" id="ID7ID9_gui">
        <omgdi:waypoint x="505" y="180"/>
        <omgdi:waypoint x="615" y="180"/>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge bpmnElement="ID3ID6" id="ID3ID6_gui">
        <omgdi:waypoint x="210" y="165"/>
```

```
                <omgdi:waypoint x="210" y="300"/>
                <omgdi:waypoint x="300" y="300"/>
            </bpmndi:BPMNEdge>
            <bpmndi:BPMNEdge bpmnElement="ID6ID7" id="ID6ID7_gui">
                <omgdi:waypoint x="400" y="300"/>
                <omgdi:waypoint x="490" y="300"/>
                <omgdi:waypoint x="490" y="195"/>
            </bpmndi:BPMNEdge>
        </bpmndi:BPMNPlane>
    </bpmndi:BPMNDiagram>
</definitions>
```

A.2 Sizes of Harmonic Template Areas

The size of the large area of V,Y and X types are 26% of the wheel, the small areas of i,L,I and Y are 5%, the large area of L is 22%, the area of T is 50%, while the angle between the centers of the two areas of I, Y and X are 180°, the angle between the centers for the two areas of L are 90° [24].

A.3 Textual Description and Model of the Demo Process

The Spark GmbH wants to analyze one of their processes. The process is a production/development process of products.

Everything starts with either a client request or a marketing idea. After that, the request must be analyzed by an expert, and then the request is compared with existing products. Then the expert must choose an existing product for use, if he has decided which product is used, then the corresponding product needs to be produced. If they want to use the "Reforge" product they have to decide where it should be produced. But if they want to use the "Recycle" product they have to produce it at production site Alpha. The "Reforge" product can be either produced at production site Beta or produced at production site Omega. After that, the product needs to be shipped to the main facility. In the main facility, the quality of the product is checked, if there are quality issues, then the process must start again before an existing product is chosen. But if the product does not have

any quality issues, the product is packed, after that the product is shipped and then the process is finished with the delivery.

A.4 How-to of the Application

universität
ulm

How to Use the Application

Contents

1 Start with the Application

After you have started the application, it should look like in Figure 1.1 and a cursor in the text field should be blinking, this means you can start inserting text into the text field.

1.1 Insert text

There are two possibilities to insert text into the text field, but first make sure focus is on the text field and you see a blinking cursor. After that you can either type in the process description you want or you past your text into the text field. It is also possible to insert text into existing text. The highlighted elements will not be influenced by this, except if you insert text into a highlighted element, then your new text will be considered part of this highlight.

1.2 Find possible BPMN element descriptions

After a process description is inserted into the text field, the application will provide hints to possible BPMN elements through the change of text color. The color corresponds with the possible BPMN elements. This marked text can be part of BPMN elements but also only provide a suggestion for some BPMN elements like splits, end-, and start-events. This can be seen in Figure 1.2

1.3 Highlight text elements

To highlight elements it is only needed to select the text which should be highlighted and press the button of the desired BPMN element. Then if you deselect the text the

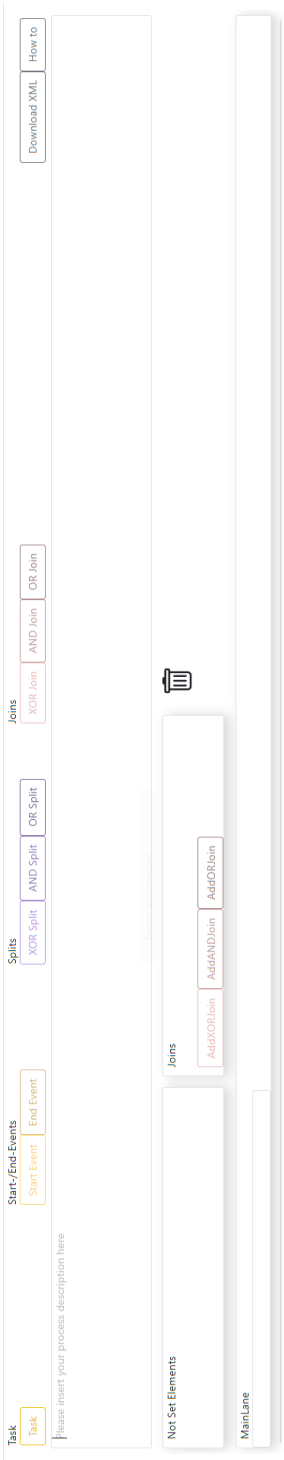

Figure 1.1: The start of the application

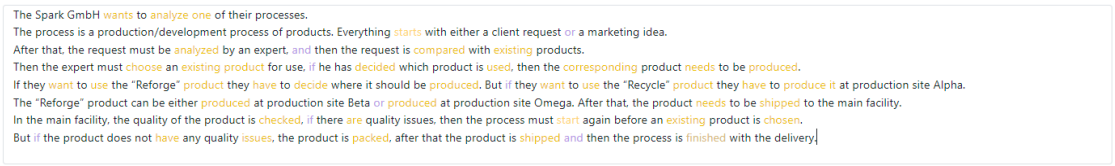

Figure 1.2: Suggestion for possible BPMN element descriptions

highlighting should be visible. The highlighting with selection of a text can be seen in Figure 1.3, where a start element is highlighted. After the highlighting, the highlighted element should also be visible in the *Non Set Elements* list or the *Joins* list. Chapter 2 explains what is possible with these elements. If you want a Join, but there is no corresponding text section for it, then it is possible to create a Join by clicking the button below of the *Joins* list of the Join you want to create without highlighting text.

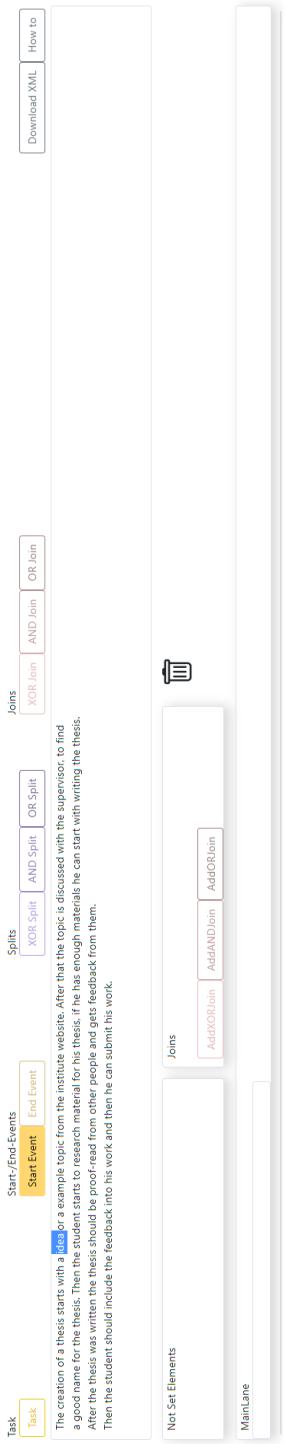

Figure 1.3: Highlight a start element

2 Use of the highlighted elements

2.1 Drag and Drop

The highlighted elements can be dragged and dropped into different lanes. Whether an element can be dropped can be seen by a slightly grey hue in the background of the lane. This can be seen in Figure 2.1. Sometimes there will be a field containing "...". Elements can be dropped on there to create a new lane. This is especially needed for splits, because they need more than one lane at a time, to for example model parallelism. The "..." lane can be seen in Figure 2.2. It is not possible to insert elements after a Join, Split or End Event.

2.2 Open/Close Splits and Joins

Sometimes it is needed to open a Split or Join to insert elements into their lanes. To do so, the Split/Join needs to be already placed in a lane, then it is possible to click onto the Split/Join and it will open up in the application as a new list. This can be seen in Figure 2.3. It is also possible to close these lists again, because only the latest two opened lists will be displayed. To close a list you only have to click onto the close button in the upper right hand corner of the list, which can be seen in Figure 2.4.

2.3 Simple Modelling Examples

Now that you know how to handle the elements, it is possible to start creating models. The elements just need to be set into the lists in the decided order. In Figure 2.6 can be seen that the *Main Lane* is closed, but the start and *decide which pizza* split was inserted in it. After that the *decide which pizza* split was opened and the decision was

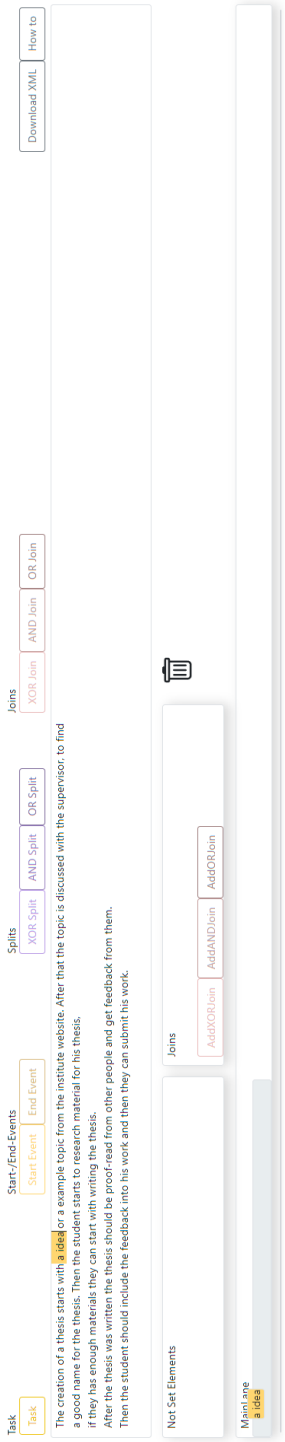

Figure 2.1: Drag and Drop a element in the *Main Lane*

Figure 2.2: The ... Lane inside of *Main Lane*

created through inserting the wanted elements and at the end the Join closes the lanes. At last the End Event had to be inserted and that is how a simple model was created, this can be seen in Figure 2.7. The textual highlighting of the elements can be seen in Figure 2.5

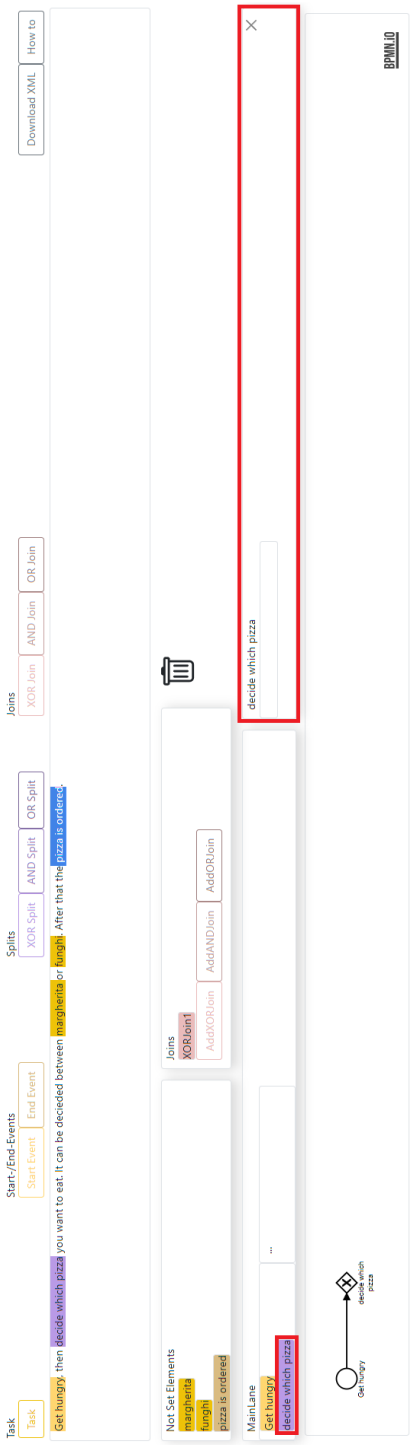

Figure 2.3: Open a Split element to insert elements into it

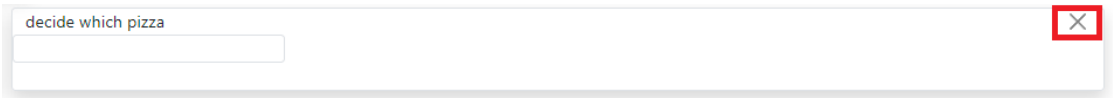

Figure 2.4: Close a opened list

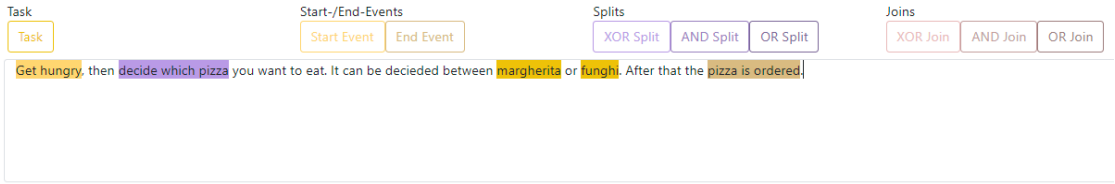

Figure 2.5: The highlighting of the simple pizza process

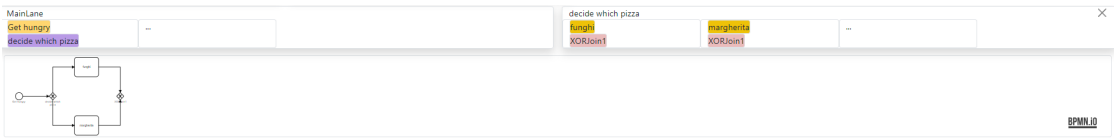

Figure 2.6: The two first lists for the pizza process

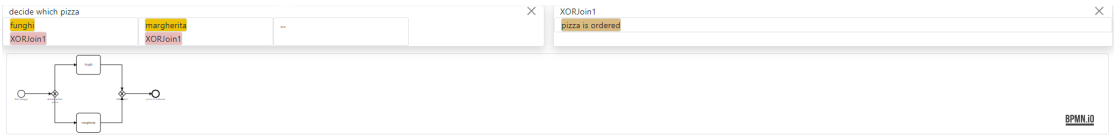

Figure 2.7: The second list and the last list needed for the modeling of the pizza process

3 Deletion of highlights and elements

There are two possible ways of deleting highlighted elements, one is the use of the garbage can symbol 3.1 and the other is to delete text 3.2.

3.1 Use of garbage

This method is quite easy, if you want to delete a element from a list, which is not the *Not Set Elements* or *Joins* list, then you easily drag the element to the garbage and it will be deleted from the lane it was in before. But if you want to delete an element completely, you need to drag it out of the *Not Set Elements* or *Joins* onto the garbage, then the element will be deleted completely including the highlighting. The dragging of a element onto the garbage can be seen in Figure 3.1.

3.2 Text deletion

The other method to delete a highlighted element completely is to delete the corresponding text that was highlighted to create the element, as changes to the text are reflected in their respective element.

Figure 3.1: Dragging of a element onto the garbage

4 Download the XML

To download the XML of the created BPMN model you have to click in the upper right side of the application on the *Download XML* button. The button can be seen in Figure 4.1.

Honesty Disclaimer

I hereby affirm that I wrote this thesis independently and that I did not use any other sources or tools than the ones specified.

Ulm, 14.December.2021
Place, Date

Kai Kretschmann
Kai Kretschmann

903179
Matriculation Number