



Enhancing ProMoEE and DyVProMo with Additional Features to Foster Empirical Studies in the Context of Process Models Comprehension

Master's Thesis at Ulm University

Submitted By:

Florian Loth

florian.loth@uni-ulm.de

994158

Reviewer:

Prof. Dr. Manfred Reichert

Prof. Dr. Rüdiger Pryss

Supervisor:

Michael Winter

2022

Version March 8, 2022

© 2022 Florian Loth

Satz: PDF- \LaTeX 2 $_{\epsilon}$

Abstract

Business Process Management (BPM) has become an important factor on management level for enterprises, as it offers the opportunity to increase productivity and lower cost. This has led to a wide use of BPM techniques in the industry, offering the ability to describe processes, improve and automate them or respond to changes quickly. In order to visually represent processes, notations are used. One of the most common is Business Process Model and Notation (BPMN), which is capable of displaying interconnected activities along with resources and other information. A process model that does not accurately represent the real world may lead to a reduction in above benefits. Therefore, enterprises have an interest in skilled experts creating high quality process models. In reality a lot of untrained personnel is involved in the modeling process. Hence, there is interest in efficient ways of helping novices to understand modeling languages. That is why research on the comprehension of process models is being conducted. One area of research is the addition of constructs to the existing notation. In particular, the coloring of modeling elements can help to distinguish and recognize them more easily.

In this thesis two pre-existing applications dealing with the assistance of conducting research on modeling comprehension are fostered. One is an application to dynamically change the displayed model elements to reduce complexity or providing help with model element names through the addition of annotations. It is fostered by expanding its functionality to dynamically add predefined colors to the model elements, providing another way of supporting understanding. The other is a survey platform with the ability to create questionnaires including the functionality to view and edit process models. Hence, it aims at the conduction of empirical research on model comprehension. It is improved in its Maintainability, usability and Applicability. Moreover, the first application is fully integrated into the second one, providing the ability to use it for surveys in questionnaires.

Acknowledgments

First and foremost, I have to thank my supervisor Michael Winter for his guidance and support in the creation process of this thesis.

I would also like to show gratitude to Prof. Dr. Manfred Reichert and Prof. Dr. Rüdiger Pryss for reviewing this work.

Furthermore, I would like to thank Florian Gallik for the assistance in supervision.

I also express my thanks to the staff of the Institute of Databases and Information Systems for the interesting content in lectures and exercises during my Master studies.

Moreover, I have to thank my friends Illari, Janosch and Julian for their encouragement and patience.

Most importantly, I thank my parents for their continued moral and financial support and my sister for her support and warm-heartedness, without them it would not have been possible for me to finish my studies.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Contribution	2
1.3	Structure of the Thesis	2
2	Fundamentals	4
2.1	BPMN 2.0	4
2.1.1	BPMN in Color	9
2.2	Perceptual Discriminability	10
2.3	Secondary Notation	11
2.4	Color Theory	12
2.5	Usability	12
2.6	Refactoring	13
2.7	Node.js	14
2.7.1	npm	14
2.8	TypeScript	15
2.9	Bpmn-js	15
2.10	CSS Frameworks	15
2.10.1	Bulma	16
2.10.2	Bootstrap	16
2.10.3	TailwindCSS	16
2.11	React	17
2.11.1	Components	17
2.11.2	Hooks	18
2.11.3	Create-React-App	18
2.12	Redux	18

2.13 Dropwizard	19
3 Related Work	20
3.1 ProMoEE	20
3.2 DyVProMo	22
3.3 Color in process models	23
4 Introduction to ProMoEE and DyVProMo	26
4.1 ProMoEE	26
4.1.1 Architecture	29
4.2 DyVProMo	30
5 Requirements Analysis	33
5.1 Functional Requirements	33
5.2 Nonfunctional Requirements	34
6 Design	37
6.1 Comparison of ProMoEE and DyVProMo	37
6.2 Addition of Secondary Notation to DyVProMo	38
6.3 DyVProMo model file import and export	40
6.4 Alignment of the projects	40
6.5 Integration of DyVProMo into ProMoEE	40
6.6 Improving ProMoEE's Usability	42
6.7 Refactoring ProMoEE	44
7 Implementation	45
7.1 DyVProMo	45
7.2 ProMoEE	49
8 Requirements Comparison	54
8.1 Functional Requirements	54
8.2 Nonfunctional Requirements	56
9 Conclusion and Outlook	58
9.1 Conclusion	58

Contents

9.2 Outlook	59
9.2.1 DyVProMo	59
9.2.2 ProMoEE	61
A Sources	63
Bibliography	69

Listings

2.1	Example BPMN 2.0 XML Shape with color definition [44]	10
2.2	Example configuration for npm	14
7.1	Coloring function for model elements	46
7.2	Color detection of file import	48
A.1	Props of the BpmnViewer with default values	63
A.2	HighlightBtn Component written in JavaScript	63
A.3	HighlightBtn Component written in TypeScript	64
A.4	CreateUser written as class Component	64
A.5	CreateUser written as functional Component	67

List of Figures

2.1	A BPMN Task element	5
2.2	An Example of a Sequence Flow	5
2.3	Example of an Exclusive Gateway, based on [47]	6
2.4	Example of a Parallel Gateway, based on [47]	6
2.5	A Start Event, a Message Intermediate Catch Event and an End Event	7
2.6	A Pool divided into two Lanes	7
2.7	A Message Flow	7
2.8	A Data Object and a Data Store	8
2.9	A Text Annotation and an Association	8
2.10	An example process of ordering a pizza [46]	9
2.11	Example experiment for visual search, based on [24]	11
2.12	Different harmonic color relationships inside hue wheels [8]	12
3.1	Process model colored in Stark et al.'s color scheme [57]	25
4.1	Overview of all available questionnaires as created by Kreßmann . .	28
4.2	Analysis view of a BPMN Modeler model task as created by Kreßmann	28
4.3	Overview of all available questionnaires with new design	29
4.4	Comparison of two BPMN models inside the ProMoEE Comparator .	29
4.5	Parts of a process model in DyVProMo with overlay explanations turned on	31
4.6	The DyVProMo application showing a process with one lane highlighted	32
6.1	DyVProMo's colors for their respective modeling element type	39
6.2	Packaging process for DyVProMo	41
6.3	Database scheme of ProMoEE	42
7.1	DyVProMo with colored elements and highlighting	47

List of Figures

7.2	Process model export buttons	47
7.3	Error message on unknown colored model import	47
7.4	Visibility setting at creation of questionnaire page type “BPMN DyVProMo”	49
7.5	DyVProMo viewer in a questionnaire	50
7.6	Questionnaire overview in new design	51
7.7	Keyboard shortcuts and full-screen buttons	52
7.8	Validity of questionnaire pages / questions (left) and input fields (right)	52
7.9	New error message dialog	52
7.10	Dates in English formatting	53

List of Tables

2.1	BPMN in Color extension attributes [44]	10
6.1	Technology usage of ProMoEE and DyVProMo	38
8.1	Comparison of functional requirements	54
8.2	Comparison of nonfunctional requirements	56

1 Introduction

This chapter represents the motivation for this thesis. Furthermore, the contribution of this thesis is provided, followed by its structure.

1.1 Motivation

Business Process Management (BPM) is concerned with techniques and methods to discover, analyze, restructure, execute and monitor business processes [59]. Given its potential benefits in lowering cost and increasing efficiency it has gained a lot of attraction in recent years [11, 68].

One way to represent such business processes is through process models, providing the ability to document, enhance and execute them. They are defined by a notation, typically in graphical form which consists of activities, their dependencies to each other, used data and resources [68]. One of the most widely used [20] is the modeling language Business Process Model and Notation (BPMN) [47]. The company-wide use of those modeling languages has led to the contact of non experts with business process modeling [3]. Being a complex matter [33], this is potentially problematic [36]. In order to utilize above benefits, the process models should reflect the real world as closely as possible. To do so, modeling experience is needed. This has led to a need in understanding the process of how models are comprehended in order to effectively train novices to do so [33]. Therefore, research is being conducted to better understand the process of understanding process models [59]. Among other things, the usage of secondary notations [21], i.e. colors, to improve understandability is investigated [13]. The use of colors offers the benefit of being readily available, as it is not being used by modeling tools, which are often using a tool-specific implementation of the modeling grammar [57].

1.2 Contribution

To support the research on process models' understanding, two applications were created as Master's theses [19, 30] before.

The first is an application to dynamically change information displayed by a process model viewer. Often times multiple people with different capabilities regarding process modeling work on the same models. A static approach that would help novices, e.g. annotating modeling elements, would hinder experts, as they overload the process model, making it harder to effectively use them. Therefore, the created application uses a dynamic approach to bridge the gap between novices and experts [19].

The second application is a web-based tool for surveys regarding BPMN 2.0 modeling. It offers the opportunity of creating, executing and analyzing questionnaires with tasks specific to process modeling. Questionnaire elements can be created that include the interactive viewing or modeling of BPMN process models. The application aims at the conduction of empirical studies in the context of process modeling and the introduction of novices to BPMN 2.0 [30].

Goal of this thesis is the fostering of those two applications to further support research on the subject of model comprehension. Therefore, the first application should be enhanced by the functionality of dynamically changing modeling elements' colors. Furthermore, the second application should be improved in its Usability, Applicability and Maintainability. Besides that, the first application should be integrated into the second, allowing its usage in questionnaires.

1.3 Structure of the Thesis

The rest of this thesis is structured as follows:

Chapter 2 provides needed fundamentals on BPMN, Perceptual Discriminability, secondary notation in the context of modeling languages, basics on color theory, Usability and refactoring, as well as details about technologies used in the implementation. Describing the research that inspired the two pre-existing applications

and other work on color in process models is Chapter 3. Chapter 4 provides a detailed introduction to those two applications. The requirements for the implementation are defined in Chapter 5. Chapter 6 describes the design of this thesis' work. The implementation is illustrated in Chapter 7. Chapter 8 is concerned with the comparison of the implementation with the requirements. Finally, Chapter 9 summarizes this thesis.

2 Fundamentals

This chapter provides needed fundamentals for the thesis. It is concerned with the business process modeling language BPMN, Perceptual Discriminability, secondary notation in the context of modeling languages, basics on color theory, Usability and refactoring, followed by the description of technologies used in this thesis' work.

2.1 BPMN 2.0

The Business Process Model and Notation (BPMN) standard was developed by the Object Management Group (OMG) with version 2.0 in 2011. Visualizing business processes as models, its goal is to bridge the gap between stakeholders in contact with business process models. On the one hand it needs enough specificity, facilitating the models' creators to visualize a wide variety of complex processes in detail. On the other hand, it should be easily understood by process executors and giving managers a good overview on the process landscape. Secondly it aims at assuring the visualization of executable business processes, defined in XML, in a business-oriented way. Therefore, the standard defines 3 diagram types: *Collaboration* diagrams, *Conversation* diagrams and *Choreography* diagrams. *Collaboration* diagrams visualize the interactions between multiple business units, their message exchange and their respective activities. *Choreography* diagrams define the expected behavior between Process participants, i.e. their message exchange. *Conversation* diagrams depict the logical relation of message exchanges [47]. Being used to model business processes, *Collaboration* diagrams are the ones used in this thesis.

Following are descriptions of the main elements used inside a *Collaboration* diagram.

Activity

Activities visualize the work that is being performed in a Process with Tasks being a subset, describing the smallest unit of work [47]. They are depicted by a rounded rectangle (cf. Figure 2.1).

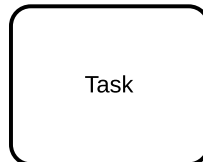


Figure 2.1: A BPMN Task element

Sequence Flow

A Sequence Flow connects Activities, indicating the order in which those are executed [47]. They are depicted by an arrow with a solid line (cf. Figure 2.2).

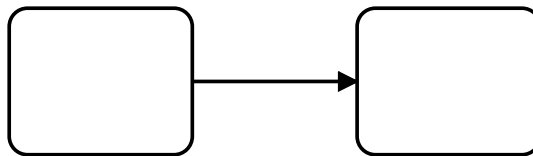


Figure 2.2: An Example of a Sequence Flow

Gateway

Gateways diverge and converge the Sequence Flow and can be used to branch, fork, merge or join paths. They are depicted by a rhombus including an inner marker, defining the type of gateway. The most frequent being *Exclusive* and *Parallel* Gateways. Exclusive Gateways diverge the Sequence Flow by creating alternative paths, only one of which can be followed (cf. Figure 2.3). Parallel Gateways split the Sequence Flow into multiple paths, executed simultaneously (cf. Figure 2.4) [47].

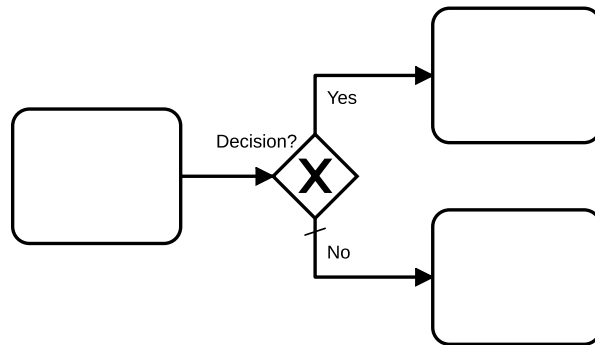


Figure 2.3: Example of an Exclusive Gateway, based on [47]

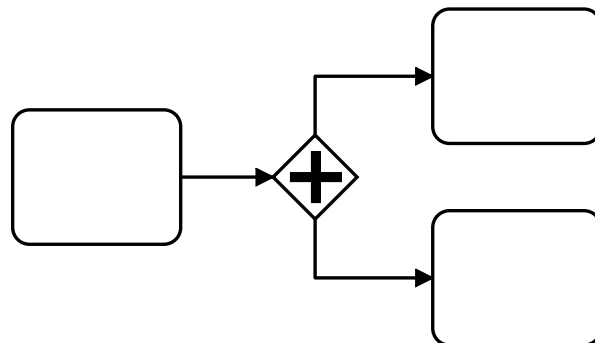


Figure 2.4: Example of a Parallel Gateway, based on [47]

Event

Events occur in a Process and affect the Process flow by triggering something or being triggered. The following three different types of Events exists: *Start*, *Intermediate* and *End* Events. Start Events mark the beginning of a specific path in a Process and therefore have no incoming Sequence Flow. Intermediate Events occur in the middle of a Process. End Events mark the end of a specific path in a Process and therefore have no outgoing Sequence Flow [47]. They are depicted by circles with Intermediate Events having an inner marker depending on the type of trigger (cf. Figure 2.5).



Figure 2.5: A Start Event, a Message Intermediate Catch Event and an End Event

Pools and Lanes

A Pool visualizes an entity partaking in a Process. Pools can be subdivided into Lanes, describing a specific role, system or department [47]. They are depicted by a rectangle (cf. Figure 2.6).



Figure 2.6: A Pool divided into two Lanes

Message Flow

Message Flows visualize the flow of messages in communication between two entities partaking in a Process [47]. They are depicted by an arrow with a dashed lined (cf. Figure 2.7).

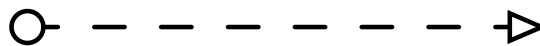


Figure 2.7: A Message Flow

Data Object and Data Store

Data Objects are objects needed during process execution. This includes physical and informational items. Using Data Stores, information can be fetched or saved for usage outside the Process [47].

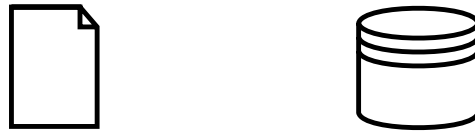


Figure 2.8: A Data Object and a Data Store

Association and Text Annotation

Associations are connections between information or Artifacts, like Data Objects or Data Stores, and Flow Objects, e.g. Tasks [47]. Text Annotations are a way of providing additional information to a model element connected via Associations. They are depicted as a dotted line (cf. Figure 2.9).

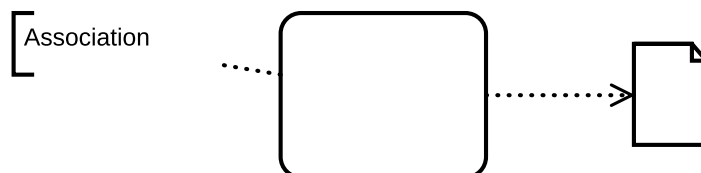


Figure 2.9: A Text Annotation and an Association

Bringing together these elements in a single process model, Figure 2.10 shows an example process of ordering a pizza. The two Participants, *Pizza Customer* and *Pizza vendor* are visualized by pools and their communication by message flows between those pools. After the *Pizza Customer* selects and orders a pizza, the order is received by the clerk of the *Pizza vendor*, who transfers the order to the pizza chef, baking the pizza. When 60 minutes have passed without the pizza arriving, the *Pizza Customer* asks the clerk for the pizza and is being calmed by them. After the pizza has finished baking, the delivery boy delivers the pizza to the

Pizza Customer, they handle the pizza's payment, after which the *Pizza Customer* can eat the pizza.

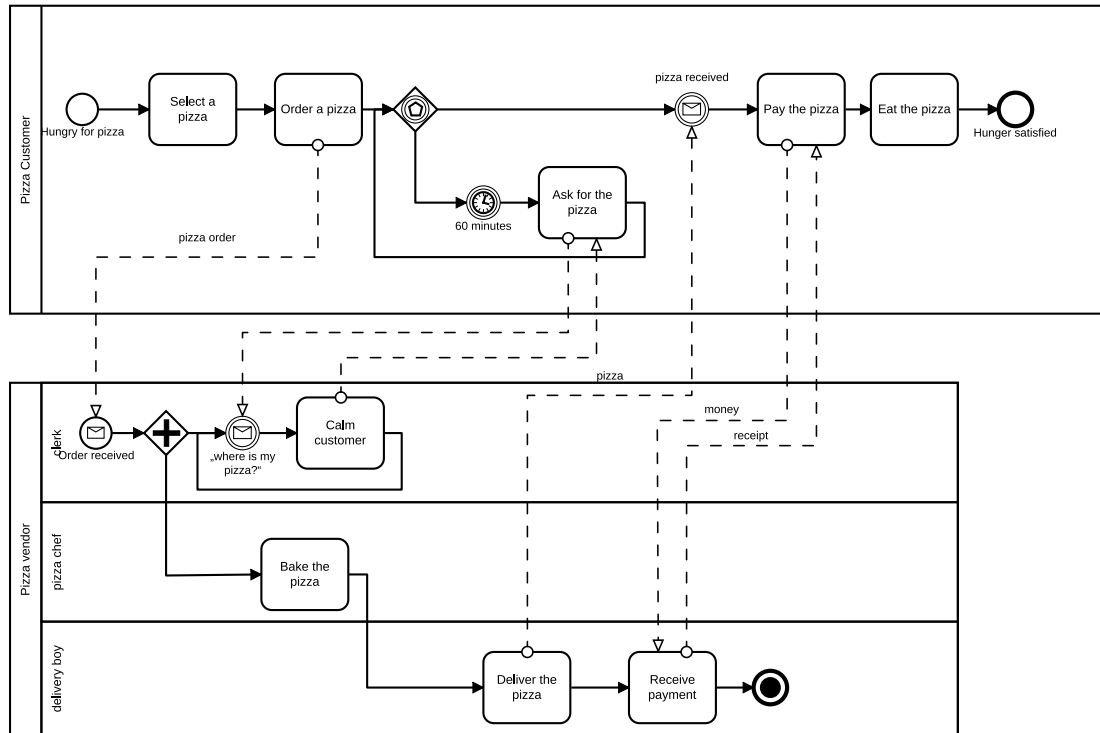


Figure 2.10: An example process of ordering a pizza [46]

2.1.1 BPMN in Color

“BPMN in Color” is a specification created by the OMG BPMN Model Interchange Working Group (BPMN MIWG) [45] in 2014. Building on top of the BPMN meta-model, it specifies the standard for defining colors inside a BPMN 2.0 file in order to interchange elements’ colors. The color extension is defined in the <http://www.omg.org/spec/BPMN/nonnormative/color/1.0> namespace. Its extension attributes are shown in table 2.1 and an example of a BPMN 2.0 XML file is shown in Listing 2.1. The colors must be provided in HEX format, leaving no option for gradients, shadows or transparency [44].

Table 2.1: BPMN in Color extension attributes [44]

Attribute Name	Description/Usage
background-color: HexColor	The background color defines the fill color of <i>BPMNShape</i> .
border-color: HexColor	The border color defines the color of the contour line of <i>BPMNShape</i> and the line color of <i>BPMNEdge</i> .
color: HexColor	The color attribute defines the color of the text depicted by a <i>BPMNLabel</i> .

```

1 <bpmndi:BPMNShape bpmnElement="task1" id="d1-task1"
   color:background-color="#ffa500" color:border-color="
   #000000">
2   <dc:Bounds x="0" y="0" width="100" height="100"/>
3   <bpmndi:BPMNLabel labelStyle="normal-text"
     color:color="#ffffff"/>
4 </bpmndi:BPMNShape>

```

Listing 2.1: Example BPMN 2.0 XML Shape with color definition [44]

2.2 Perceptual Discriminability

Perceptual Discriminability describes the “ease and accuracy with which graphical elements can be differentiated from each other” [40]. It can be used to differentiate model elements from each other or to highlight (pop-out) specific elements [55]. Multiple frameworks exist [32, 40, 51], providing principles on how to develop a notation with *Perceptual Discriminability* in mind. Those works are based on visual search, which focuses on the impact of discriminators, like different kinds of shapes in combination with colors, through the conduction of studies [55].

An example experiment, conducted by Healey et al. is provided in Figure 2.11. A target (red circle)—only being present in pictures b, d and e—needs to be located in between surrounding elements of different shapes and colors.

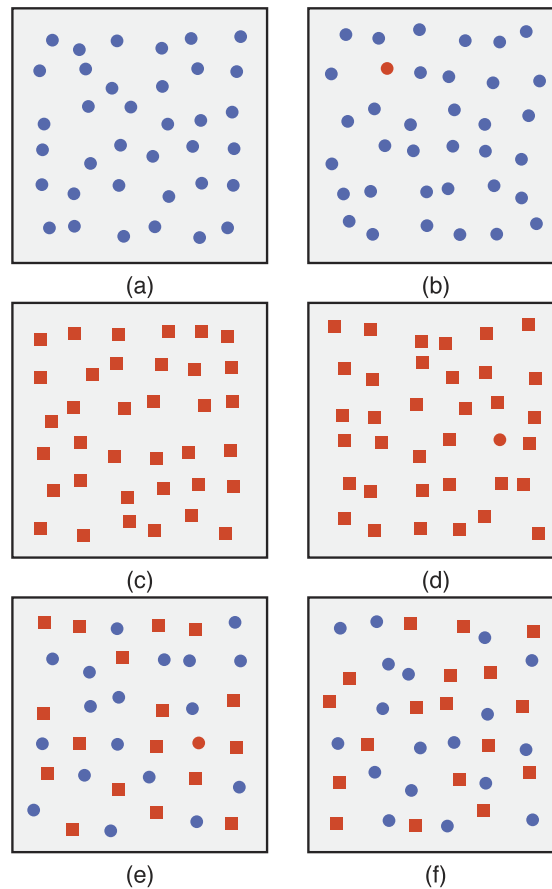


Figure 2.11: Example experiment for visual search, based on [24]

2.3 Secondary Notation

Targeting the factors that determine understanding of a process model, a lot of research has been conducted, rather focusing on *how* the process is represented, not *what* it represents. This research can be split up into *primary* and *secondary notation* [21]. The former is concerned with particular symbols or the shape of modeling elements and—by that—their meaning. In turn, the latter focuses on layout, colors or annotations [31] to “reinforce or clarify meaning” [40] by expressing structures, otherwise not available [21].

Changes in primary notation are often hindered by tool-specific adjustments from the tools’ vendors, making it a very slow process [54]. Research solutions or changes in secondary notation, however, can be implemented more quickly [21,

57].

2.4 Color Theory

One important part, when thinking of color selection in the context of this thesis, is legibility, which is determined by contrast between text and background [34].

Another part is color harmony, meaning a group of aesthetically pleasing colors. Theories exist on how to choose these sets of colors [35, 63, 65]. With colors aligned in a hue circle, specific relationship between them should produce a harmonic palette. A set of possible relationships is shown in Figure 2.12. It is possible to combine those with a variation in lightness and saturation to create a desired effect [34].

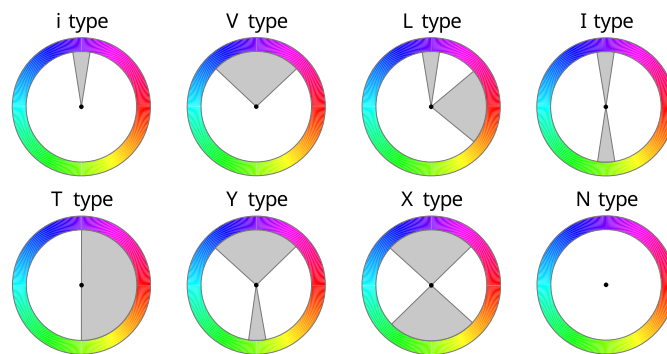


Figure 2.12: Different harmonic color relationships inside hue wheels [8]

2.5 Usability

Although playing an important role in human computer interaction, there does not exist a single definition for the term usability that is being agreed upon [10].

For simplicity purposes, the definition given by Nielsen in [43] will be used for the scope of this thesis. According to him, usability consists of the following five components:

- *Learnability*: Learning the application should be easy, making it possible to start working fast.
- *Efficiency*: Using the application should be efficient, making it possible to reach a high degree of productivity after learning it.
- *Memorability*: The application should be easy to memorize, making it possible to return to using it after time of absence, without the need of relearning everything.
- *Errors*: The application should have a low number of errors, minimizing the need to recover from those. Additionally, catastrophic errors must not occur.
- *Satisfaction*: Using the application should be subjectively satisfying to the user.

Further, “less is more” is one of Nielsen’s slogans of importance for this thesis [43]. By that he is referring to the false need of designers to include a multitude of options or features into an application.

For a broader overview, a summary of definitions can be found in [10].

2.6 Refactoring

During the continued development of a software project, it is modified and changed according to new or suspended features. This enlarges the code base, making it deviate from the original design, probably lowering the quality. Therefore, the need for techniques to cope with this trend exists, which were given the name “refactoring” [37].

Refactoring in Computer Science is defined as “the process of changing a software system in a way that does not alter the external behavior of the code yet improves its internal structure” [18]. Further, the result of refactoring is a software system, maintaining a good design throughout the whole development process [18].

Although constraints in resources often hinder engineers of performing refactoring, studies show its benefits, e.g. in lowering defects [26] or improving Maintainability and reusability [37].

2.7 Node.js

Node.js is an open source JavaScript runtime environment to run JavaScript code—outside of a browser—on a server. In recent years it received a wide adoption with over 1 million downloadable modules for it. Thanks to this popularity, a variety of different applications can be built using it, e.g. HTTP web servers, microservices or command-line applications [22].

2.7.1 npm

Npm is the default package manager of *Node.js* and is used to share JavaScript software (packages). To do so, developers upload them to a public database, the so-called *registry* [66]. The *registry* can be browsed and shared packages can be downloaded.

In order to download and use the provided packages, a project maintainer can specify the wanted package inside the project's configuration file. This file is called *package.json*. In it, dependencies to other packages are defined by stating the unique name of the package, followed by the version to download from the network. Additionally, packages can be imported locally by stating a file path to the package, instead of a version. An example configuration is displayed in Listing 2.2 which shows the syntax for both—network and local—dependency inclusion types that are defined inside the configuration file's dependency section [67].

Furthermore, npm is the command-line application to handle the needed interactions with the *registry* in order to share packages and maintain projects.

Listing 2.2: Example configuration for npm

```
1 "dependencies": {  
2   "react": "~17.0.2",  
3   "dyvpromo": "file:../../dyvpromo-0.1.0.tgz"  
4 }
```

2.8 TypeScript

The TypeScript language and compiler is a transpiler for JavaScript with the addition of static type support, developed by Microsoft in 2014 [39]. TypeScript builds on top of JavaScript, retaining that syntax while adding its own. Types are checked before runtime, “making TypeScript easier to debug and test than JavaScript” [12]. It includes the TypeScript compiler (tsc), capable of transpiling TypeScript code into JavaScript code [39].

2.9 Bpmn-js

Bpmn-js is a JavaScript library for interaction with BPMN 2.0 diagrams, developed by Camunda Services GmbH [6]. It provides the functionality to integrate BPMN viewers or modelers into other JavaScript projects and is split into three different parts:

- bpmn-moddle
- diagram-js
- bpmn-js

On the one hand, *bpmn-moddle* handles the interaction between BPMN 2.0 XML files and JavaScript, making it possible to read and write those files. On the other hand, *diagram-js* provides the user interface, visualizing the BPMN diagrams and supplying the user interaction methods to view or model the diagrams. Finally, *bpmn-js* is the connection of the above two. It builds the bridge between them and provides a single API for the library [6].

2.10 CSS Frameworks

Cascading Style Sheets (CSS) are used to define the presentation of HTML documents [38]. With development becoming increasingly complex, CSS frameworks have gained popularity to simplify and streamline the development process [62].

Those frameworks provide a set of predefined CSS files, filled with selectors that follow a specific overall style or design. The framework maintainers decide on the naming of those selectors. Therefore, with no chance of avoiding possible naming conflicts, using multiple CSS frameworks will lead to an unexpected behavior, as similar named selectors will override each others values.

Following, the three frameworks used in the projects of this thesis are described.

2.10.1 Bulma

Bulma [61] is a lightweight CSS library, as it uses a single CSS file. It provides the opportunity to use several User Interface (UI) elements in order to style a web page. Those include: column features for layouting, boxes, buttons, notifications, progress bars, drop-down menus, navbars and forms. They are harmonized in size, shape and spacing accompanied by a set of balanced colors (cf. Section 2.4). Therefore, *Bulma* does not offer a lot of flexibility, but makes it easy to create a visually appealing UI with little effort or design knowledge that works on mobile devices as well.

2.10.2 Bootstrap

Bootstrap [4] is a CSS framework that consists of a collection of CSS and JavaScript files to build the UI for a web page. Apart from static UI elements, used for the page's design, interactive elements to control those elements are provided as well. The framework provides the possibility to either use the elements as is, or to customize them to a great amount [29]. Further, a large selection of templates, themes, additions and extensions exists, from which a user of the framework can choose. It also works for desktop and mobile devices in the same manner [29].

2.10.3 TailwindCSS

TailwindCSS [60] is a CSS framework that focuses on flexibility. Providing mostly small wrappers for CSS styling, it is very similar to writing plain CSS code, but

written using its own syntax inside an HTML file. Although offering a larger flexibility, this comes at the cost of a lot more code to write and the need for a higher degree in CSS knowledge [49].

2.11 React

React (or ReactJS) [16] is a JavaScript library to build UIs for web pages developed by Facebook. It uses a single-page approach, meaning that the page's content gets downloaded completely while being updated on interaction without using the browser's normal workflow of fetching new documents or reloading them when following hyperlinks. This aims at replicating the feeling of native applications—either mobile or desktop [7].

In order to break a UI down into manageable parts, React uses JavaScript functions, the so-called *Components* which are described by the following section. By doing so, React is enabled to update only the specific parts that have changed when displaying changes to the screen, making it efficient [12].

2.11.1 Components

Components define the elements of the UI in a React application [12]. The following three types of Components exist, each having their own set of features:

- Functional Components
- Class Components
- Factory Components [12]

As their name states, *functional Components* are JavaScript functions, introduced to React in August 2015 [25]. However, *class Components* are based on JavaScript classes, typically used to create objects [41]. Inheriting their functionality from a React class, they were the standard of how to build a Component, before Hooks were introduced by Facebook (cf. Section 2.11.2).

Being deprecated from React version 17 on forward, *factory Components* are not further discussed.

2.11.2 Hooks

Among other drawbacks [15], class Components introduce a lot of boilerplate code [7]. Therefore, Facebook introduced Hooks in 2018 [15], bringing a new way of handling Components' state. Hooks are only applicable inside functional Components [15].

Although focusing on this new way of handling state, Facebook states, that the old way using class Components will not be deprecated [15].

2.11.3 Create-React-App

Create-React-App is the official [14] and most widely used [12] way to bootstrap a React application. It provides a multitude of templates (e.g. JavaScript or TypeScript) to create a fully functional application from. This application consist not only of React but also includes an orchestrated set of additional tools like a build tool, a testing library and scripts to run the application locally, all without the need of configuring them manually. Therefore, making the initial creation of a React application very quick and easy [12].

2.12 Redux

The creators of Redux define it as “a predictable state container for JavaScript apps” [1]. It enables the management and thereby the centralization of an application's state [12]. Being available for standalone JavaScript, a specific version for React [2] exists.

In the context of React, Redux performs the task of a global state object that is accessible from within the whole application. It shares the state between Components, without them having to share the state variables (called local state) [12].

2.13 Dropwizard

Dropwizard is a Java framework for developing RESTful web services. It consists of a multitude of Java libraries to do so, some of them are: the *Jetty* HTTP server, the mapper *Jersey* to map between HTTP requests and classes, the JSON library *Jackson* and the relational database accessor *JDBI* [9].

3 Related Work

This chapter describes related work of this thesis, including the research that inspired the two applications being fostered in this thesis, as well as various research on color in process models.

3.1 ProMoEE

ProMoEE is a surveying and questionnaire tool for the specific context of BPMN process modeling, developed by Kreßmann during a Master's thesis at Ulm University in 2019 [30]. It supports empirical research on the topic of understanding process models by serving as platform for the creation, execution and analysis of surveys and questionnaires on BPMN process modeling. The ability to view and model process models is integrated into the survey workflow. This platform is one foundation for this thesis, as it is being enhanced further. A more detailed description of the application's functionality and its state before this thesis is provided in chapter 4.

The preexisting application, influencing the creation of *ProMoEE* is *PEX* [23]. Developed as part of a Bachelor's thesis by Gutermuth in 2016 at Ulm University, it supports the planning, execution and evaluation of experiments regarding the optimization of processes using BPMN models. Experiments help to detect and verify potential process optimizations and efficiency gains in business processes. Implemented as a web application, experiments can be created, shared between experiment creators and sent to participants. Those are guided through the experiments, answering questions or creating models on the basis of given statements or questions. Furthermore, experiment creators have the ability to evaluate the par-

ticipants' created models. Another feature of PEx is the ability to create alternating work flows, making it easy to conduct control group experiments [30].

In turn, *PEx*'s inspiration was the *Cheetah Experimental Platform* (CEP), developed by the University of Innsbruck in 2010. It is a desktop application also supporting experimental research on business process modeling to gain more valuable insights compared to paper based experiments or qualitative data. According to Pinggera et al., *CEP* has several benefits compared to paper based experiments. In the experimental design phase, it helps to define objects, subjects and the execution order of different tasks. Additionally, it contains the tools that are often being used in experiments, like surveys, tutorials and process modeling tools. During experimental execution its straightforward workflow keeps participants on the experiment's path, assuring the designed setup is being followed, leading to higher data validity. Lastly, more sophisticated data analysis is provided through tools measuring metadata like time spent on different tasks or model replaying abilities. As *PEx* does, it supports the creation of alternating work flows for control group experiments. Centerpiece of *CEP* is the so-called "Cheetah Modeler", providing the aforementioned abilities of model creation based on a given task and the later analysis of these models, including replaying the exact modeling process [48].

Kreßmann, the creator of *PromoEE*, states as reason for the new implementation its focus on the execution of surveys and questionnaires, as well as non-functional requirements, like performance and stability. Additionally, as a web application, lowering the effort by only requiring a browser, the potential participant number and ease of use for experiment creators is increased [30].

Further inspiration was taken from the application *QuestionSys*, initially developed during a diploma thesis at Ulm University by Scherle in 2014 [53]. Aiming to support psychologists at conducting validated questionnaires, a group of mobile applications has been developed, providing the tools to design, validate, execute and analyze them in a process-driven manner [64].

3.2 DyVProMo

The second foundation for this thesis is the application *DyVProMo*, also developed during a Master's thesis at Ulm University by Gallik in 2021 [19]. *DyVProMo* was created to provide a tool for dynamic visibility changes of BPMN process models' elements, aiming at improving empirical research of model comprehension, like *ProMoEE* does. A more detailed description of the application's functionality is also given in chapter 4.

In a systematic literature review Stein Dani et al. observe the current research status¹ on the “visualization of business process models” in the context of improving perception and comprehension [59]. Therefore, they adopt two angles. The first being the focus on similarities between the reviewed papers in order to classify them into the following categories: “Augmentation of existing elements”, “Creation of new elements”, “Exploration of the 3D space”, “Information visualization”, “Visual feedback concerning problems detected in process models” and “Perspectives”. Secondly, they analyze the papers by a visualization analysis framework leading to an overview of the suggestions made by those. They found that user interaction features are scarcely researched on, however half of the research is based on the BPMN standard. Their conclusion states that, with BPMN being an ISO standard and widely adopted, more research is needed on that topic with their work being an inspiration to do so [59]. As the literature review focuses on existing work, no proposition of a visualization technique is being made by the authors.

ProView is a framework developed at Ulm University to change process views of process-aware information systems (PAIS) in order to lower the work load on domain experts maintaining the process models. Being abstractions of larger, more complex models, process views help to provide information for a distinct use case by abstracting the large model in a specific manner (e.g. a manager needs a broad overview, while a worker needs a very detailed part of the process model regarding their specific work). *ProView* helps to maintain those large process models by allowing changes in the process views to be propagated to the model. In order to control those changes, the model has to be linked to its process views, accompanied by a set of instructions ensuring the correct execution of changes [27, 28].

¹publishing dates between 2009–2018

The specificity of *ProView* and need to configure each model on its own prevents a general application for all process models.

3.3 Color in process models

Firstly, research on *primary notation* includes the connection of routing symbols—or Gateways—to the process model comprehension [17, 50].

Secondly, the following examples for research on *secondary notation* are concerned with syntax highlighting, layout and annotations, aiming at model comprehension as well.

Reijers et al. propose a formal approach to syntax highlighting in workflow nets, coloring matching operator pairs in the same color. They further implement this approach and conduct a thorough study on its effectiveness. Their conclusion states that it has a high usefulness for novices and is applicable to other process modeling languages as well [52].

La Rosa et al. create a number of layout patterns for process models. Some of these patterns are: language specific *layout guidance*, *enclosure highlighting* of specific elements in a rectangular box, *graphical highlighting* of model elements (especially coloring), *pictorial annotations* like icons or images, *textual annotations*. After verifying the support by various modeling languages they conduct a usability evaluation of the patterns with BPM practitioners [32].

Natschläger proposes a solution called *Deontic BPMN* which transforms BPMN process models for a better readability by lowering their complexity. Based on path exploration, a model is potentially simplified and tasks following an exclusive gateway are given a specific color according to their mandatory state. This minimizes the required amount of process knowledge when executing a process [42].

In [31] Kummer et al. focus on cultural differences in color perception. Conducting a study they detect a difference between two diverse cultures and provide implications for a culture dependent color scheme when coloring process models for understandability. Further they state, cultural values should be taken into account when conducting research on color in process models [31].

The work of Erol determines that the coloring choices are mostly left to the modeler and are often poorly supported by tools. Arguing that, for an effective use of color to support understandability an aesthetically pleasing set of colors needs to be used, he proposes a variety of color schemes, developed based on color theory and creates a prototypical implementation of those [13].

Similarly, in a series of work [55, 56, 57, 58] Stark et al. created a color scheme for BPMN process models, based on *Perceptual Discriminability*. Firstly, they extend the latter for application in process model language design [55, 56]. Secondly, in a systematic approach including theories of color vision, color harmony and visual attention, they create two balanced color schemes aiming at the minimization of visual stress [58]. Consolidating their findings in [57], where they begin with the identification in published literature of four scenarios regarding which modeling constructs need to be scanned in order to answer questions for a given model. Those scenarios include: *Process flow only*, *Tasks and Process flow*, *Tasks, Process flow and Pools*, *Pools and tasks* [57]. Further, they determine the following free visual variables in the BPMN 2.0 standard, not contradicting primary notation: Brightness, Hue and Saturation. Combining those findings with the necessity to emphasize particular modeling elements in order to pop-out [40], they develop a color scheme for BPMN 2.0 (cf. Figure 3.1) [57].

The work by Stark et al. represents the theoretical groundwork for the coloring functionality developed in this thesis.

The above selection shows a multitude of research on colors to improve model comprehension, while agreeing on the benefits of using colors [13, 31, 57], a lot of effort is put into finding a suiting color scheme, balancing the benefits of Perceptual Discriminability and the introduction of visual stress. In trying to find the perfect balance they propose different fixed changes to the process model, none are using a step-wise, dynamic approach. Therefore, this thesis adopts the different scenarios identified in [57] to visualize them in a dynamic way in order to support further research on the topic of model comprehension.

4 Introduction to ProMoEE and DyVProMo

This chapter provides detailed information about the two projects that are being enhanced in this thesis. It includes the process of their development, their functionality, as well as their respective technologies.

4.1 ProMoEE

As already stated in Section 3.1, *ProMoEE* (short for Process Modeling Experimental Editor) was initially developed by Kreßmann during his Master's thesis in 2019 at Ulm University. It is a web application to conduct empirical research in the field of process model comprehension through surveys and questionnaires.

On completion of the thesis, a questionnaire could consist of at least one page with one of the following types:

- Question
- BPMN Viewer
- BPMN Modeler

A *BPMN Modeler* is a standalone questionnaire page that provides a BPMN 2.0 process modeling tool. No questions or other elements can be displayed on that page. The model for participants to start with can be created during the questionnaire creation process. Questionnaire participants have to answer one line of statement or question with a business process model.

A *BPMN Viewer* questionnaire page shows a BPMN process model on top that can be zoomed in and out or panned interactively, followed by a minimum of one question.

Questions are typical questionnaire question elements that either are part of a *BPMN Viewer* or a questionnaire page consisting only of questions. They can have one of the following types:

- Text input,
- Text area,
- Single choice,
- Multiple choice.

The application supports the creation and management of these questionnaires. Users are able to view their created questionnaires and share, edit or delete them. The overview of their questionnaires, shown in Figure 4.1, is presented to every user. Questionnaires are shared by link with which a participant is eligible to fill it in.

Completed questionnaire answers can be viewed and analyzed by questionnaire creators. The latter process is supported by statistics for each participants' created model, e.g. the number of tasks and the prettiness of the sequence flows is shown (cf. Figure 4.2). Furthermore, the answers can be exported as Excel file. In order to manage users, an admin user is able to create and delete those.

Following the Master's thesis of Kreßmann, the application was further developed by students within the framework of two projects. During those, *TailwindCSS* was introduced to parts of the application, resulting in a new design (cf. Figure 4.3). Moreover, a new question type was created: the scale. Also, a new questionnaire page type was introduced: the *BPMN Comparator*, which allows putting two process models side by side while highlighting differences between them.

In order to reconstruct the modeling steps of a *BPMN Modeler* page, a modeling history was introduced, allowing questionnaire creators to see each modeling step a participant made. Furthermore, the ability to upload process models during questionnaire creation was added.



Figure 4.1: Overview of all available questionnaires as created by Kreßmann

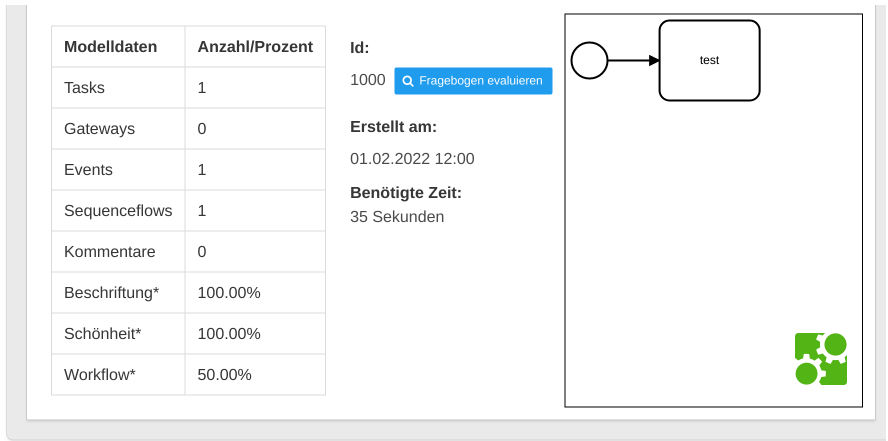
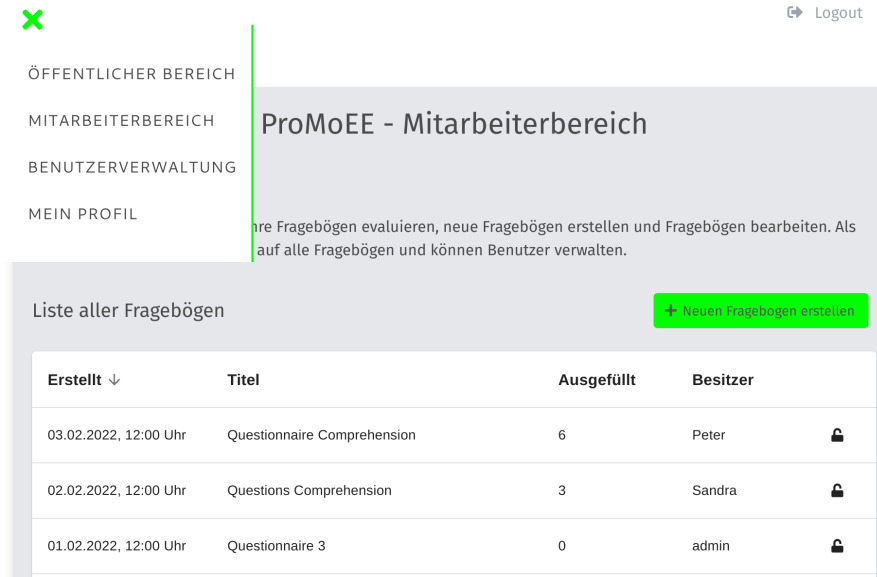


Figure 4.2: Analysis view of a BPMN Modeler model task as created by Kreßmann



The screenshot shows the 'ProMoEE - Mitarbeiterbereich' interface. On the left is a sidebar with navigation links: 'ÖFFENTLICHER BEREICH', 'MITARBEITERBEREICH', 'BENUTZERVERWALTUNG', and 'MEIN PROFIL'. The main area has a header 'ProMoEE - Mitarbeiterbereich' and a description: 'Ihre Fragebögen evaluieren, neue Fragebögen erstellen und Fragebögen bearbeiten. Als auf alle Fragebögen und können Benutzer verwalten.' Below this is a section 'Liste aller Fragebögen' with a green button 'Neuen Fragebogen erstellen'. A table lists three questionnaires:

Erstellt ↓	Titel	Ausgefüllt	Besitzer
03.02.2022, 12:00 Uhr	Questionnaire Comprehension	6	Peter
02.02.2022, 12:00 Uhr	Questions Comprehension	3	Sandra
01.02.2022, 12:00 Uhr	Questionnaire 3	0	admin

Figure 4.3: Overview of all available questionnaires with new design

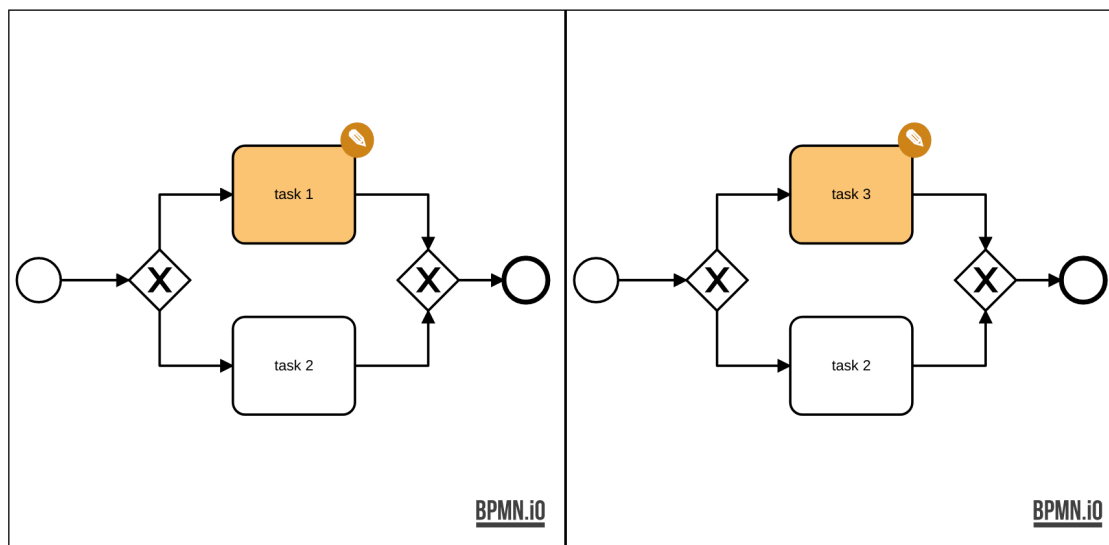


Figure 4.4: Comparison of two BPMN models inside the ProMoEE Comparator

4.1.1 Architecture

The ProMoEE project consists of two parts: the *front-end*, responsible for displaying the UI and the *back-end*, responsible for data delivery and persistence. Hence, the project's architecture is a Client-Server architecture.

The front-end was created using the Create-React-App tool (cf. Section 2.11.3) and therefore uses React and Node.js. TypeScript was chosen as language and Redux for state management.

The back-end service was developed using the Dropwizard framework. This can also be broken down into two pieces. Firstly, the RESTful API which is responsible for exchanging the data via HTTP protocol between Client and Server. And secondly, the database which is accountable of persisting the application's data (especially questionnaire and answer data).

4.2 DyVProMo

The creation of *DyVProMo* (short for Dynamic Visualization of Process Models) was part of Gallik's Master's thesis in 2021 at Ulm University. The application allows to dynamically change and highlight elements of a BPMN 2.0 process model in the context of process model comprehension.

It consists of two views: the start-page and a BPMN viewer. On the start-page users have to upload the intended BPMN model file, either via drag and drop or by clicking the upload field. A successful upload leads them to the main screen—the BPMN viewer. Here, the model is displayed, accompanied by the tools to dynamically change it [19].

The *Detail-Slider* to change the visibility of model elements is positioned in the bottom center. It has 4 distinct positions, each with a predefined setting for the visibility of Annotations, Data Object, Data Stores and overlay explanations. Overlay explanations are small text boxes hovering aside an element, containing its name. Each explanation exists once for each included element type of the model [19]. Figure 4.5 displays this behavior. Following the different levels and their respective changes are described.

- Level 1: Data Objects, Data Stores, Annotations and overlay descriptions are not visible.
- Level 2: Data Objects and Data Stores are visible; Annotations and overlay descriptions are not visible.
- Level 3 (default): Data Objects, Data Stores and Annotations are visible (effectively no changes to the model); overlay descriptions are not visible.
- Level 4: Data Objects, Data Stores, Annotations and overlay descriptions are visible [19].

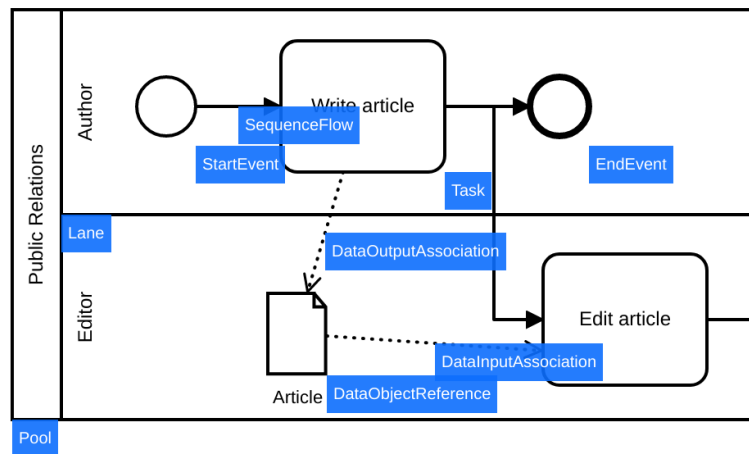


Figure 4.5: Parts of a process model in DyVProMo with overlay explanations turned on

Apart from its own positioning, the slider level is displayed as a table describing each level's visibility changes, while highlighting the current level. This supports the users understanding of the different detail levels [19].

Additionally, *visibility checkboxes*, to change the elements' visibility independently of the slider setting, are located on the top left. Apart from the slider's elements, the visibility of the Message Flows is controllable. A new setting of the detail slider will override the checkbox state for those element types affected by the new slider position. Gallik states, that the reason for the Message Flows' exclusion in the detail slider is their essential role for understanding the model's sequence flow, e.g. a Message Flow starting a process. However, as they have the ability to overcrowd the

model, a possibility to toggle their visibility, which is provided this way, must be given [19].

The ability to highlight Pools and Lanes is given by the *Highlight* tool, consisting of one checkbox for each Pool or Lane. It is located on the top right. Toggling a checkbox will highlight the background of the respective element in green color, depicted in Figure 4.6.

The exit button on the top right closes the BPMN viewer and brings users back to the start-page.

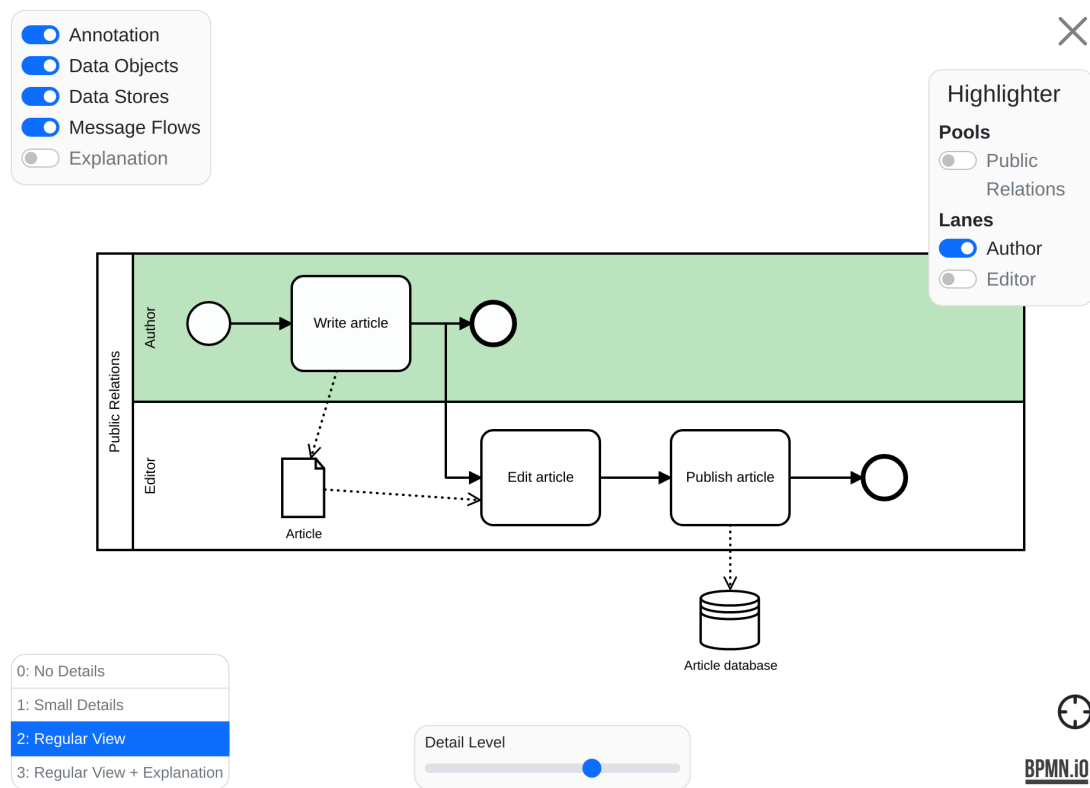


Figure 4.6: The DyVProMo application showing a process with one lane highlighted

Architecture

The application was built using the Create-React-App tool (cf. Section 2.11.3) with JavaScript. Hence, it uses the React library with Node.js.

5 Requirements Analysis

This chapter is concerned with the requirement analysis, listing the needed requirements that need to be met in order to successfully ensure the function of the system. It is split into *functional* and *nonfunctional* requirements, which in turn are split into the two projects—*DyVProMo* and *ProMoEE*. They are abbreviated accordingly: “FRD” for functional requirement *DyVProMo*, or “NFRP” for nonfunctional requirement *ProMoEE*.

5.1 Functional Requirements

FRD1: Dynamic secondary notation

DyVProMo should be able to display secondary notation for BPMN 2.0 diagrams on a dynamically changeable basis. The type of secondary notation should be colored model elements.

FRD2: Highlighting functionality still intact

The above colors should not affect the already implemented highlighting feature.

FRD3: Export process models

DyVProMo should be able to export the displayed process models as they appear on the screen. The file formats to export should be; (1) BPMN 2.0 XML file and (2) SVG vector graphics file. Potentially existing colors of the model should be stored in the exported files.

FRD4: Detect color in imported models

DyVProMo should be capable of detecting the color of imported models. Hereby a distinction between colors that *DyVProMo* is capable of displaying and other (unknown) colors should be made.

FRP1: DyVProMo in questionnaire

ProMoEE should be able to create questionnaire pages displaying the *DyVProMo* viewer. Participants then should be able to use a fully functional *DyVProMo* viewer as part of a questionnaire.

FRP2: Visibility of DyVProMo interaction elements in ProMoEE

The visibility of *DyVProMo*'s interaction elements should be adjustable during questionnaire creation.

FRP3: Applicability

ProMoEE should be utilizable for conducting surveys.

FRP4: English language support

ProMoEE should be available in the English language, besides German.

5.2 Nonfunctional Requirements

NFRD1: Integration capability

DyVProMo should be capable of being integrated into other projects.

NFRD2: Standalone application

The capacity of being integrated should not affect *DyVProMo*'s functionality as standalone application.

NFRD3: Visibility of interaction elements

The visibility of interaction elements should be adjustable by applications integrating *DyVProMo*.

NFRD4: Usability DyVProMo

The Usability of *DyVProMo* should be high according to the definition by Nielsen [43] (cf. Section 2.5).

NFRP1: Alignment

In order to integrate *DyVProMo* into *ProMoEE*, the projects should use the same underlying technology.

NFRP2: Usability ProMoEE

ProMoEE should offer a high degree of Usability.

- *Learnability*: Learning how to create and manage questionnaires should be quick.
- *Efficiency*: Creating and managing questionnaires should be efficient.
- *Memorability*: The creation and management of questionnaires should be easy to memorize.
- *Errors*: Errors should be minimized while using the application. If Errors occur, they should be well visible and instructive.
- *Satisfaction*: Using the application should be subjectively satisfying to the user.

NFRP3: Maintainability ProMoEE

The Maintainability of *ProMoEE*'s code base should be high. New maintainers should be capable to acquaint themselves with the project quickly.

6 Design

This chapter describes the design of this thesis' work: the implementation of enhancements to the two existing applications *DyVProMo* and *ProMoEE* in the context of model comprehension. Firstly, their technologies are compared to each other. After that, the enhancements of *DyVProMo* are discussed. Then the steps needed to integrate it into *ProMoEE* are illustrated. Finally, the enhancements of the latter are shown.

6.1 Comparison of ProMoEE and DyVProMo

Chapter 4 already describes the applications' functionality. As they are integrated into each other, they need to be aligned before that (cf. Section 6.4). For a more detailed technical overview, this section provides their used technologies and compares them to each other.

Similarities

Both projects are single-page web applications, created with the use of *React*. They equally were initially created using the Create-React-App tool. Therefore, they both employ a Client-Server architecture utilizing *Node.js*. Hence, a user only needs a browser in order to run one of the applications, making them independent of operating systems.

Differences

While *ProMoEE* uses class components, *DyVProMo* uses the newer functional components with hooks. The former utilizes *Redux* for a centralized state management, e.g. to handle application wide login state. The latter uses *React*'s default: local state. Another difference is the used programming language, with *ProMoEE* using TypeScript and *DyVProMo* using JavaScript. Furthermore, they both use different CSS frameworks: *DyVProMo* uses Bootstrap, *ProMoEE* uses Bulma and TailwindCSS.

In addition to the front-end, *ProMoEE*'s project includes a back-end service, handling the application's data, primarily user and questionnaire data. A continued network connection is needed in order to use the application effectively, sending and receiving data between client and server. *DyVProMo* on the other hand uses the server to solely serve the Front-End application. After opening the application, the client's browser can be used offline henceforward, still capable of importing and viewing locally stored BPMN model files.

Comparing the two applications, Table 6.1 juxtaposes their characteristics.

Table 6.1: Technology usage of ProMoEE and DyVProMo

	ProMoEE	DyVProMo
Programming language	TypeScript	JavaScript
Front-End library	React	React
State management	Redux	local state
Component type	Class Components	Functional Components
Used CSS-Framework	Bulma, TailwindCSS	Bootstrap
Back-End present	yes	no

6.2 Addition of Secondary Notation to DyVProMo

Adding to its initial functionality, dynamically changing the displayed model's elements, *DyVProMo* is being enhanced to support the dynamic adjustment of secondary notation—in this case color (FRD1).

To help with process model comprehension, the process models' elements can be colored in several stages. The colors are different for every model element type and are based on the work of Stark et al. [57]. The gradient suggested by them is omitted, as the export function (cf. Section 6.3) is only capable of using HEX values which cannot display those. In order to cover all flow objects and swimlanes, the two missing element types of other than Parallel and Exclusive Gateways and Events are allotted colors, as well. Following the approach of Stark et al. [57], those colors are in harmony with the existing ones, ensured by choosing colors analogous to them (cf. "V type" of Figure 2.12). Being more frequent, Events are colored in blue color, as the sensitivity to it is much lower compared to red or green [34].

The user is able to change those colors dynamically by choosing a level on a slider. The different levels are three of the four scenarios identified by Stark et al. [57], leading to the following colored element levels:

- **Level 1:** No element is being colored, view the model as it is (default)
- **Level 2:** Color all Gateways and Events
- **Level 3:** Color all Gateways, Events and Tasks
- **Level 4:** Color all Gateways, Events, Pools and Lanes

Those colors and their respective element type are shown in Figure 6.1.

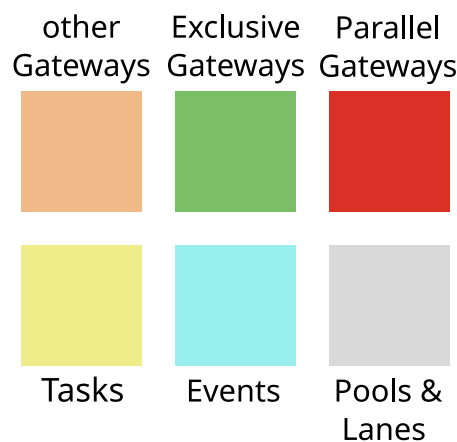


Figure 6.1: DyVProMo's colors for their respective modeling element type

6.3 DyVProMo model file import and export

Based on the BPMN in Color specification [44] (cf. Section 2.1.1) the currently viewed model can be exported as displayed on screen (including colors), either as BPMN XML file or as SVG vector graphics file (FRD3). The bpmn-js library (cf. Section 2.9) handles the export of colors according to the standard. Other modeling tools, that also support it, will be able to view the colors embedded in the XML file.

Moreover, on a model import the colors are detected, and the slider is positioned on its respective position. If an imported model contains colors other than the ones used by the application (cf. Figure 6.1), a notification is shown to the user (FRD4).

6.4 Alignment of the projects

In order to effectively integrate *DyVProMo* into *ProMoEE*, the differences between them (cf. Section 6.1) need to be minimized by aligning the projects (NFRP1). Firstly, this is achieved through updates of the shared dependencies, especially React. Furthermore, *DyVProMo* is converted to Typescript, as types improve the development process of the integration (cf. Section 2.8). However, as *DyVProMo* is integrated into *ProMoEE* and uses the default local state, there is no need to change the state management in either of the projects.

6.5 Integration of DyVProMo into ProMoEE

This section explains the steps required to integrate *DyVProMo* into *ProMoEE*. The former has to be adapted in order to make it capable of being integrated into other projects (NFRD1). Subsequently, the latter has to be adapted in order for the integration to work in questionnaires.

Changes to DyVProMo

To create the possibility of being used in other projects, the TypeScript compiler (tsc) (cf. Section 2.8) is used to compile the source code into plain JavaScript files. In order for other projects to reference those files as dependency, they have to be packaged. This is handled by the `pack` command of `npm`, which bundles the project files into a gzipped tar file. Afterwards, this file can be used as parameter for `npm`'s `install` command to add *DyVProMo* while handling the dependency resolution. This process is illustrated in Figure 6.2.

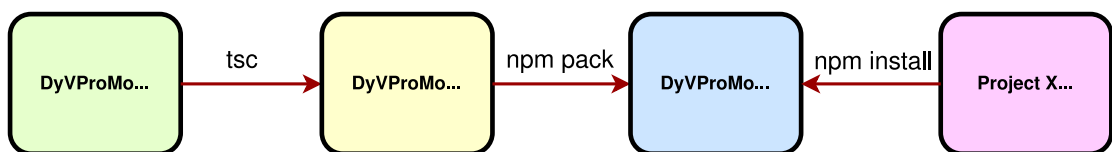


Figure 6.2: Packaging process for DyVProMo

To fulfill requirement NFRD3, the ability to programmatically set the visibility of *DyVProMo*'s tools is added. Furthermore, getting and setting of those tools' state from a using entity is added as well.

Despite those changes, as initially intended, *DyVProMo* can still be used and further developed as standalone application (NFRD2).

Integration into ProMoEE

Now it is possible to integrate *DyVProMo* into *ProMoEE* (FRP1). It is integrated as new questionnaire page type *BPMN DyVProMo* which is similar to the *BPMN Viewer*, as it allows participants to view process models while having to answer at least one question of the *Question* type.

When creating such page, the tools' visibility for the *DyVProMo* viewer is settable (FRP2). This enables questionnaire creators to test the different types of *DyVProMo*'s tools in isolation. In order to do so, the state of the tools has to be persisted on questionnaire creation. This alters the database scheme. The new scheme, including the new table *DyvpromoToolState*, is displayed in Figure 6.3.

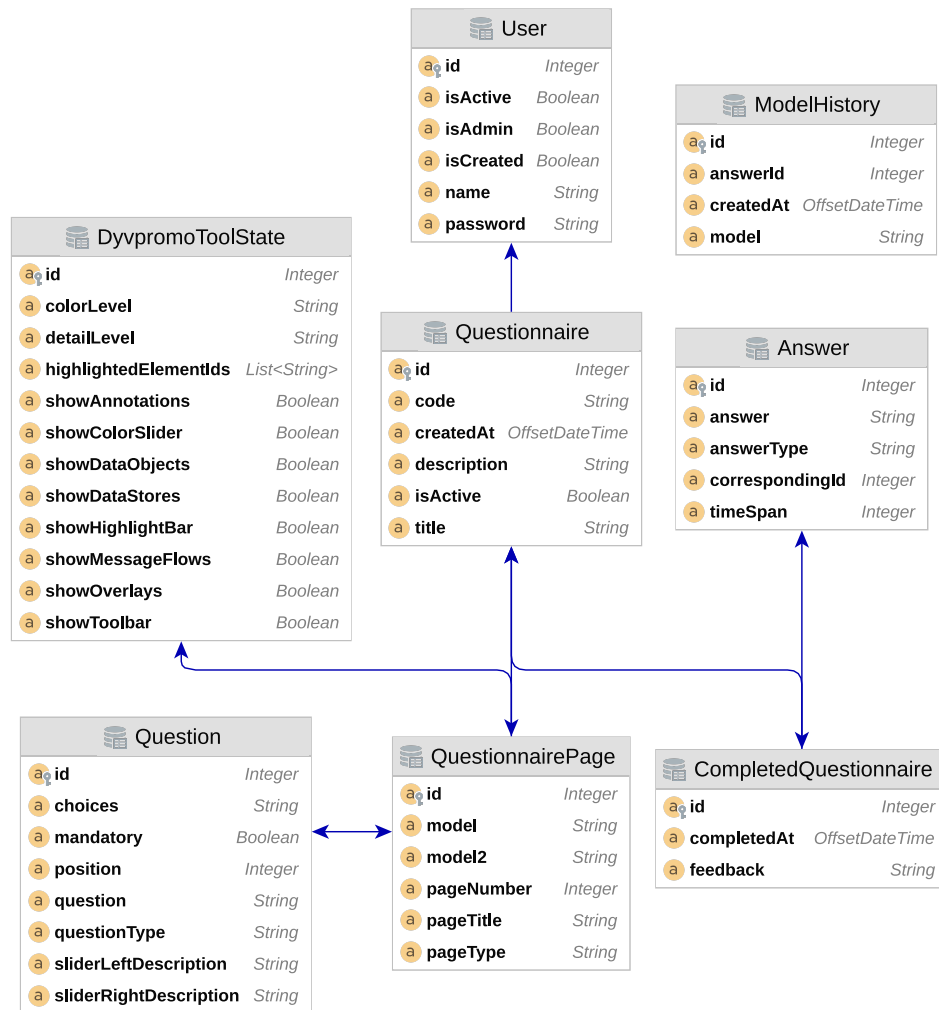


Figure 6.3: Database scheme of ProMoEE

6.6 Improving ProMoEE's Usability

When first introduced to *ProMoEE*, the usability was perceived to be moderate. Given the multiple project stages (cf. Section 4.1), features were implemented without emphasizing consistent design and usability. Therefore, improving those was made a key part of this thesis' work.

The two biggest identified weaknesses were the questionnaire creation process and the interaction with bpmn-js' integrated tool.

When creating a questionnaire, every step needed to be saved, making it an error-prone process, as it could easily be forgotten. Further, minimal requirements in text length for input fields and number of questions per page type exist that were only communicated when trying to save or preview the questionnaire. This led to another problem: the error messages when one of the above requirements was not met, were only capable of showing one message a time. This, in turn, led to backtracking the error, potentially followed by another error that had to be navigated to, making the questionnaire creation a cumbersome process.

The interaction with the bpmn-js tool was limited to the basic core of its potential functionality—for viewer and modeler alike. There was no option to zoom the displayed model. Given the restricted space (415 pixels in height), this limited the possible size of a model by a large amount, with larger models being very “movement intensive”. Moreover, during modeling there was no way of undoing the recent modeling steps.

Furthermore, notifications were placed in between other UI elements, pushing successive elements down during occurrence, resulting in an unpleasant looking effect.

In an effort to improve those issues, the following steps are taken to improve the usability (NFRP2), create a better modeling experience and overall increase the Applicability of the project (FRP3). Firstly, removing the need to save during the questionnaire creation process lowers the error rate of losing changes by a large amount. Furthermore, providing feedback for obligatory minimal requirements of input length for input fields upfront, giving an overview of met requirements for questionnaire pages and collecting potential error messages in one list lower the error rate and increases the *Efficiency* and *Satisfaction* while using it. This is also the case for the added copy button on Comparator page creation, making it easier to create differences between process models (cf. Figure 4.4). Additionally, the minimal question requirement is handled by automatically adding questions on page creation. Moreover, introducing features from the demo of the bpmn-js project [5] like keyboard shortcuts, zooming the model and the ability to view the modeling tool full-screen increase the *Efficiency*, *Memorability* and *Satisfaction*. By removing of one used CSS framework (cf. Section 6.7), using only the left one—Bulma—a more consistent design that is built to work on mobile devices, is achieved, resulting in better *Learnability* and *Memorability*. Lastly, hovering notifications lead to more

Satisfaction using the application.

6.7 Refactoring ProMoEE

To create a better code Maintainability (NFRP3), the whole application is refactored to use functional components (cf. Section 6.1), hence new maintainers only have to learn one component type (as *DyVProMo* already uses them). Furthermore, unnecessary or deprecated libraries are removed for the same reason. For example an XML parser used for BPMN 2.0 files during model analysis is replaced by *bpmn-moddle* of *bpmn-js* (cf. Section 2.9). Another example is the substitution of a deprecated date parsing library that was replaced by JavaScripts own date parsing function, leading to better maintainable code, as maintainers working with the application should already be familiar with the JavaScript documentation (FRP4).

7 Implementation

This chapter describes the implementation of the requirements in detail. It provides the changes made to the respective application, whether being technical or changes to the UI.

7.1 DyVProMo

In this section the changes made to the *DyVProMo* application are described. Those include the addition of the secondary notation, exporting of process models and recognizing colors during model import and the control over its tool state.

Secondary Notation

To enable further research on model comprehension, the ability to dynamically change secondary notation, i.e. color, is added.

A function is added to color model elements, using the coloring functionality provided by the bpmn-js library. This function is described in Listing 7.1. It takes two arguments, a list of model elements and the color as String in HEX format to color them in. Further, it passes this information to the *Modeling* entity of the bpmn-js viewer, which handles the coloring.

To control the coloring of respective elements, a slider comprised of multiple levels is added, as described in Section 6.2. Fulfilling the usability requirements set by initial creator [19], it is the same slider design that users are already familiar with, providing *Learnability* and *Memorability* of NFRD4. Figure 7.1 shows the *DyVProMo* application with the slider in the bottom center. Its color level is set to 4, coloring

the displayed model accordingly. It further shows the still intact highlighting feature highlighting lane “clerk” in green color (FRD2).

Listing 7.1: Coloring function for model elements

```
1  const colorElement = (elem: ModelElement[], color:
    string) => {
2      [...]
3
4      if (filter.length) {
5          modeling.setColor(filter, {
6              fill: color,
7          });
8      }
9  };
```

Export of process models

The feature to export the currently viewed process model, either as BPMN 2.0 XML or SVG, is added. On export the currently displayed colors are embedded in the file (FRD3) as specified by the BPMN in Color specification [44] (cf. Section 2.1.1). Figure 7.2 shows the added buttons in order to do so.

The file import is enhanced, detecting colors that were exported with above process and setting the color slider to the appropriate level according to the colors contained in the model.

Listing 7.2 shows the detection part executed during file import. The coloring is checked for each element of the given type. If all elements contain the given color, it sets the internal state (`setColorLevel`) which is read by the coloring unit which in turn sets the color slider’s level. Additionally, it checks if the model contains a different color than the expected one. When that is the case, an error message is shown to expect possibly unintended behavior (cf. Figure 7.3).

7 Implementation

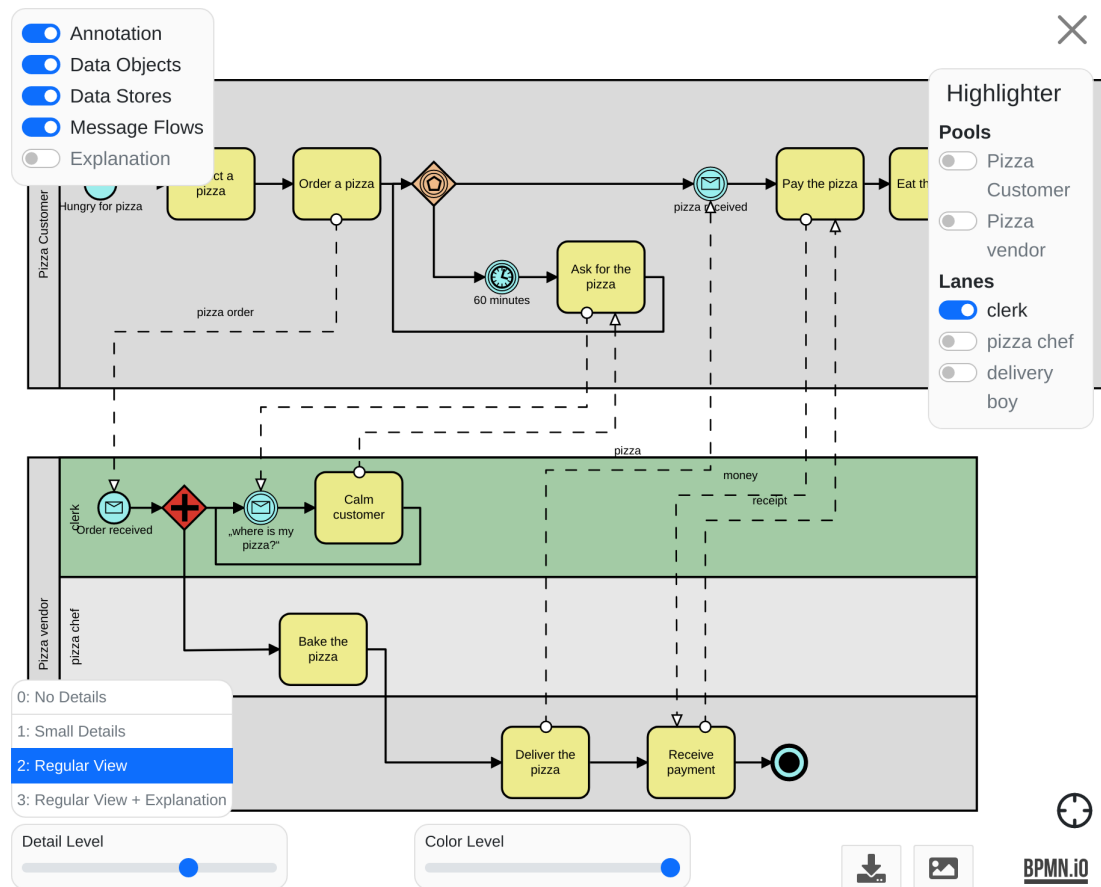


Figure 7.1: DyVProMo with colored elements and highlighting



Figure 7.2: Process model export buttons

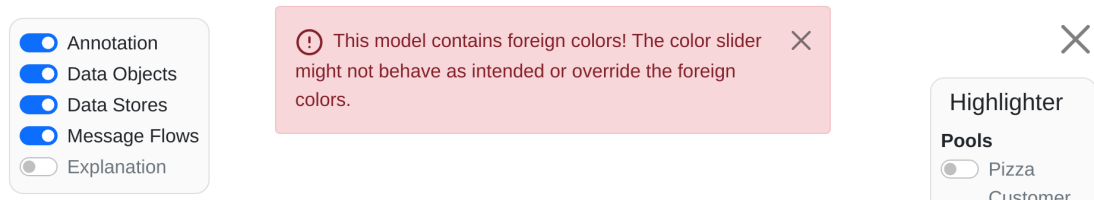


Figure 7.3: Error message on unknown colored model import

Listing 7.2: Color detection of file import

```
1 const setInitialColorLevel = (  
2   color: string,  
3   level: SliderLevel,  
4   elements: ModelElement[]  
5 ) => {  
6   if (elements.some((e) => e.businessObject.di?.["  
7     background-color"]))) {  
8     if (  
9       elements.every(  
10        (e) => e.businessObject.di?.["background-color"]  
11          === color  
12      )  
13    ) {  
14      setColorLevel(level);  
15    } else {  
16      setShowForeignColoringAlert(true);  
17    }  
18  }  
19 };
```

Controlling tool state

To integrate *DyVProMo* effectively, communication from and to the main component's (BpmnViewer) tool state is added. The application's standalone functionality is not affected, as those communication parameters are given default values or can be NULL (cf. Appendix Listing A.1).

Switch to TypeScript

Aligning the projects, the whole application was rewritten using TypeScript. Exemplifying this, the same component (HighlightBtn) is shown in JavaScript (cf.

Appendix Listing A.2) and TypeScript (cf. Appendix Listing A.3), with the major difference being the type declaration for the *props* on the top, providing useful information about their types for development.

7.2 ProMoEE

In this section the enhancements made to the *ProMoEE* application are described. Those include the integration of *DyVProMo*, the improvement of its Usability and Refactoring for maintainability purposes.

Integration of DyVProMo

DyVProMo is integrated into *ProMoEE* as a new questionnaire page type. As with the *BPMN Viewer* type, a minimum of one question needs to be added to the questionnaire page. Creators of those are able to set the visibility of *DyVProMo*'s tools (cf. Figure 7.4), allowing to test the different functions in isolation. Furthermore, the slider positions and highlighted elements are saved, making the tool appear for the participants as seen by the creator. Figure 7.5 shows *DyVProMo* as part of a questionnaire.

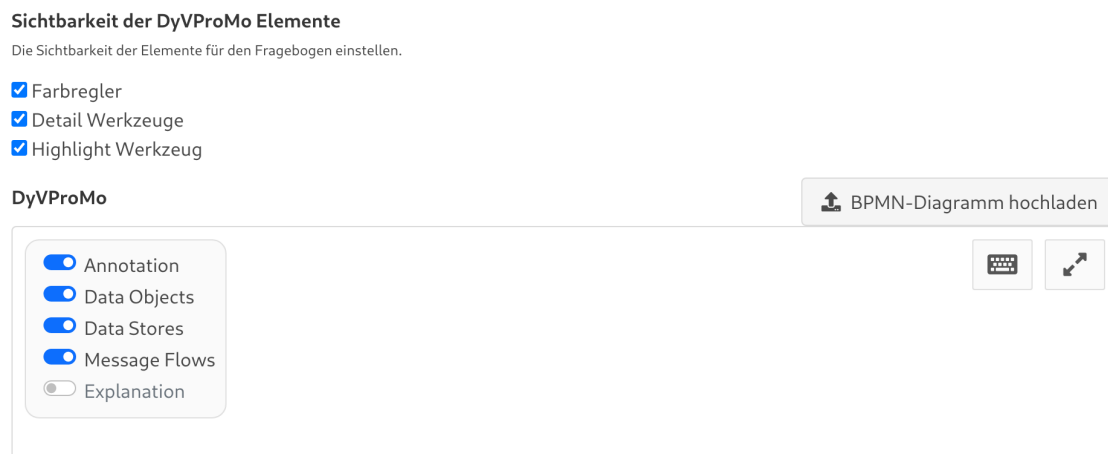


Figure 7.4: Visibility setting at creation of questionnaire page type “BPMN DyVProMo”

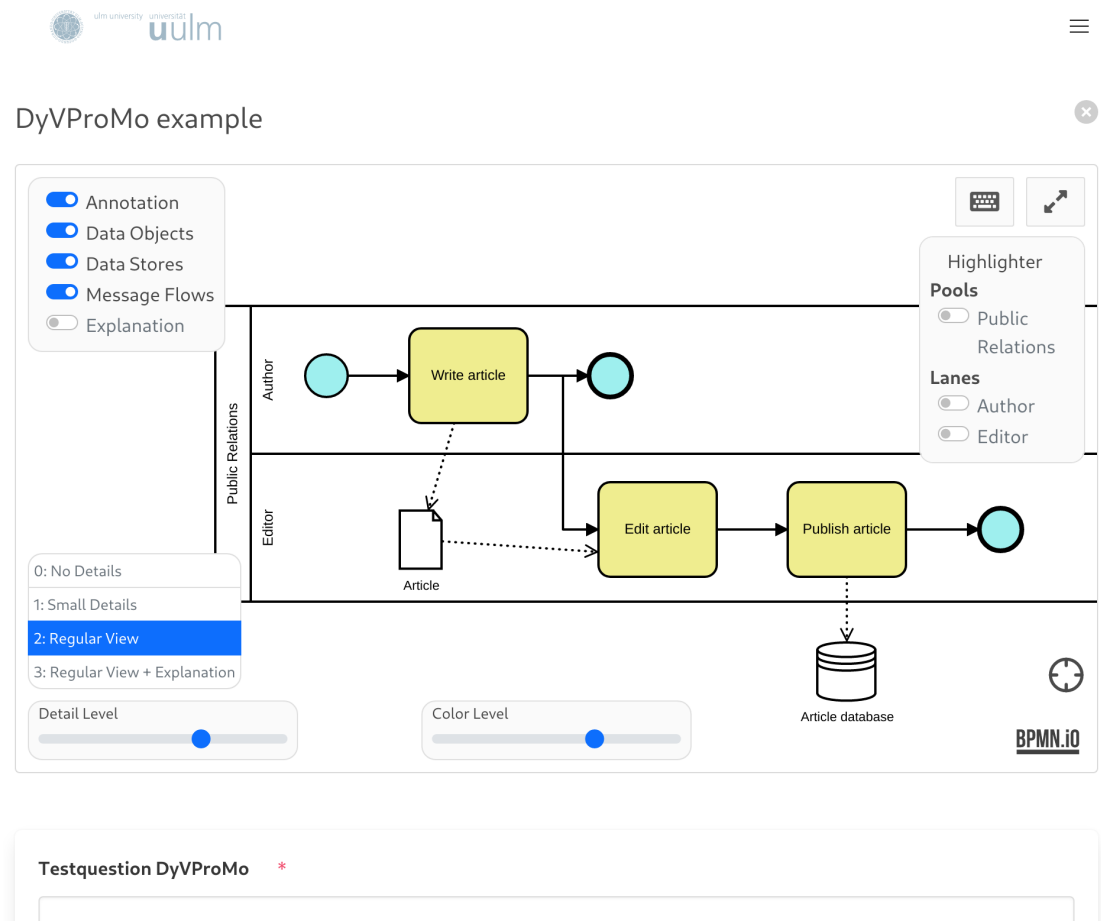


Figure 7.5: DyVProMo viewer in a questionnaire

In order to be displayed properly, *DyVProMo* needs the Bootstrap library, which is incompatible with the already present Bulma library. This is solved by providing the CSS file *bootstrap.min.css* which consists of only the Bootstrap CSS classes being used by *DyVProMo*.

Improving Usability

In removing the TailwindCSS framework (cf. Section 6.6), a more consistent, modern design is achieved which is depicted in Figure 7.6.

The user experience when creating a questionnaire is drastically improved. Changes to the questionnaire are saved automatically, making the creation process more ef-



Figure 7.6: Questionnaire overview in new design

ficient. Moreover, the added shortcuts are explained by clicking on a button which is present in every viewer and modeler of the application, fostering *Learnability*. Next to this button, also in every viewer and modeler a button that resizes the modeling area to full-screen, exists. Those buttons are shown in Figure 7.7.

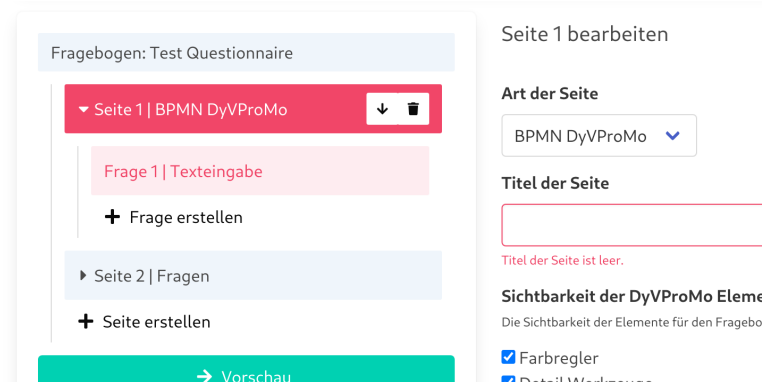
The validity of questionnaire pages and questions is shown in the questionnaire overview, providing the minimal requirement of input length for input fields upfront, giving clearer instructions to the user and minimizing *Errors*. Each questionnaire page and question has an internal state (`isValid`), which is checked and if all requirements are met, the background of the respective element in the overview turns from red to blue, indicating that this page is valid. This behavior is shown in Figure 7.8. Additionally, the error messages hinting unmet minimal requirements combines the potential errors of the whole questionnaire, providing better feedback to the user (cf. Figure 7.9).

Further, notifications throughout the whole application are taken from being integrated into the applications elements to being hovering. This leads to a more mod-

ern design, a better *Learnability* because of the known paradigm of hovering notifications from mobile devices and a higher *Satisfaction* when using the application caused by the lack of repositioned UI elements. Figure 7.6 shows such a notification in the top right corner of the screen.

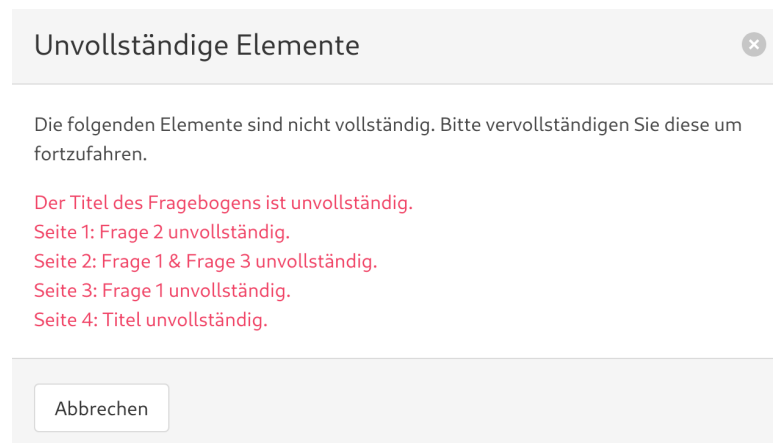


Figure 7.7: Keyboard shortcuts and full-screen buttons



The screenshot displays the application interface for editing questionnaire pages. On the left, a sidebar titled 'Fragebogen: Test Questionnaire' shows a list of pages: 'Seite 1 | BPMN DyVProMo' (highlighted in red), 'Frage 1 | Texteingabe', and 'Seite 2 | Fragen'. Below the list are buttons for '+ Frage erstellen' and '+ Seite erstellen'. At the bottom of the sidebar is a green button labeled '→ Vorschau'. On the right, the 'Seite 1 bearbeiten' panel is visible. It includes a dropdown for 'Art der Seite' (set to 'BPMN DyVProMo'), a text input field for 'Titel der Seite' (which is empty and has a red border with the message 'Titel der Seite ist leer.'), and a section for 'Sichtbarkeit der DyVProMo Elemer' with checkboxes for 'Farbregler' and 'Detail Med...'.

Figure 7.8: Validity of questionnaire pages / questions (left) and input fields (right)



The screenshot shows an error message dialog titled 'Unvollständige Elemente' with a close button (X) in the top right corner. The message text reads: 'Die folgenden Elemente sind nicht vollständig. Bitte vervollständigen Sie diese um fortzufahren.' Below this, a list of incomplete elements is shown in red text: 'Der Titel des Fragebogens ist unvollständig.', 'Seite 1: Frage 2 unvollständig.', 'Seite 2: Frage 1 & Frage 3 unvollständig.', 'Seite 3: Frage 1 unvollständig.', and 'Seite 4: Titel unvollständig.'. At the bottom of the dialog is a button labeled 'Abbrechen'.

Figure 7.9: New error message dialog

Refactoring

In order to create a better Maintainability, the whole application was rewritten using functional Components. This simplifies the learning effort for new maintainers, as they only need to learn one Component type. The difference in code is shown in the Appendix' Listing A.4 and Listing A.5.

With the removal of the deprecated *MomentJS* library, date parsing is handled by JavaScript's own parser (`Intl.DateTimeFormat`). Adding the benefit of parsing dates in the correct format according to the user's browser language. An example of a parsed date for the English language is shown in Figure 7.10.

6		42
Datum ↓	Beantwortete Fragen	Beantwortet
2/14/2022, 10:04:03 PM	7	0
2/12/2022, 8:09:40 PM	7	1
2/7/2022, 7:41:56 PM	7	1

Figure 7.10: Dates in English formatting

8 Requirements Comparison

This chapter compares the requirements set in Chapter 5 with the work of this thesis shown in Chapter 7. It is subdivided into functional requirements and nonfunctional requirements.

8.1 Functional Requirements

The following table shows the implementation status of the functional requirements.

Table 8.1: Comparison of functional requirements

Code	Requirement	Implementation status
FRD1	Dynamic secondary notation	implemented
FRD2	Highlighting functionality still intact	implemented
FRD3	Export process models	implemented
FRD4	Detect color in imported models	implemented
FRP1	DyVProMo in questionnaire	implemented
FRP2	Visibility of DyVProMo interaction elements in ProMoEE	implemented
FRP3	Applicability	implemented
FRP4	English language support	partially implemented

FRD1: Dynamic secondary notation

Implemented. *DyVProMo* can dynamically display colors for different model elements.

FRD2: Highlighting functionality still intact

Implemented. The coloring of the elements does not interfere with the coloring from the already existing highlighting functionality.

FRD3: Export process models

Implemented. *DyVProMo* is capable of exporting colored BPMN 2.0 XML and SVG files.

FRD4: Detect color in imported models

Implemented. *DyVProMo* can detect colors in imported model files and set the color slider accordingly. If a model includes unknown colors, the user is notified.

FRP1: DyVProMo in questionnaire

Implemented. *DyVProMo* is integrated into *ProMoEE*, enabling to create questionnaires with a *DyVProMo* viewer as part of them.

FRP2: Visibility of DyVProMo interaction elements in ProMoEE

Implemented. The visibility of *DyVProMo*'s tools can be set on questionnaire creation.

FRP3: Applicability

Implemented. The modelers and viewers of *ProMoEE* have a broader functionality and bugs were fixed, ensuring a higher Applicability.

FRP4: English language support

Partially implemented. Because of time-constraints the translation of the application could not be implemented, only the localization of dates was.

8.2 Nonfunctional Requirements

The following table shows the implementation status of the nonfunctional requirements.

Table 8.2: Comparison of nonfunctional requirements

Code	Requirement	Implementation status
NFRD1	Integration capability	implemented
NFRD2	Standalone application	implemented
NFRD3	Visibility of interaction elements	implemented
NFRD4	Usability DyVProMo	implemented
NFRP1	Alignment	implemented
NFRP2	Usability ProMoEE	implemented
NFRP3	Maintainability ProMoEE	implemented

NFRD1: Integration capability

Implemented. *DyVProMo* can be packaged and used as dependency in other projects.

NFRD2: Standalone application

Implemented. *DyVProMo* can still be used and further developed as standalone application.

NFRD3: Visibility of interaction elements

Implemented. *DyVProMo*'s tool visibility can be set programmatically.

NFRD4: Usability DyVProMo

Implemented. *DyVProMo*'s Usability is not affected, as the newly integrated tools follow the same design principles as the existing ones.

NFRP1: Alignment

Implemented. Both projects now use TypeScript with functional Components on the same React version.

NFRP2: Usability ProMoEE

Implemented. *ProMoEE*'s Usability was vastly improved.

- *Learnability*: New consistent design with hovering notifications.
- *Efficiency*: Creation of questionnaires is now substantially more efficient.
- *Memorability*: Modern design and auto saving make it easier to use.
- *Errors*: Errors while creating a questionnaire are minimized by auto saving and more concise information about required input from the user.
- *Satisfaction*: The new design and above error reduction make it more satisfactory to use.

NFRP3: Maintainability ProMoEE

Implemented. The removal of multiple dependencies and the usage of functional components lead to a lowered learning effort when starting to work with the projects.

9 Conclusion and Outlook

This chapter concludes the thesis. Moreover, the second section provides an overview of potential future functionalities for the applications that have been enhanced in this work.

9.1 Conclusion

This thesis evolves two pre-existing applications developed during Master's theses and integrates one into the other. Both of them have the purpose of supporting research on process model comprehension.

The first application is capable of viewing BPMN 2.0 process models and dynamically changing their displayed elements in order to improve the understanding of them. It is enhanced by the inclusion of a secondary notation. In detail this means the ability to dynamically add color to the displayed elements to further increase understanding of the model. Each element type is given a distinct color to easier distinguish them from each other. Those colors are meticulously selected based on recent research to ensure a high increase in understanding while keeping the visual distraction low. The colors are adjustable by a slider with various setting options, each of which is coloring a specific set of elements. Furthermore, the application was extended to export the process models including the colors as set, offering the possibility to exchange the colored models files with other supporting tools. Additionally, the files' colors are recognized on import and the slider position is set accordingly.

The second application is a web-platform assisting the creation and conduction of surveys regarding Business Process Modeling. Therefore, it provides the function-

ality to create, fill and analyze questionnaires. Their elements include the interactive viewing or modeling of business process models. For analysis, the ability to replay the creation process of such modeling tasks, exists. It is enhanced regarding its Usability, Applicability and Maintainability. The questionnaire creation process is made more error resistant and the design of the entire application is updated, increasing its Usability. Furthermore, the functionality of the integrated modeling tools is improved by the addition of keyboard shortcuts and a full-screen ability, further increasing Usability and Applicability. Additionally, the code base is refactored to use fewer dependencies and making it easier to learn.

To further enhance the functionality of the second application, the first one is included into it, providing the possibility to use the process model viewer, including the newly added secondary notation, in questionnaires. To ensure flexibility doing so, the various tools of the first application can be changed in visibility for a to be created questionnaire. The integration can be achieved because both applications use the React library.

With the provision of these tools, the possibility is offered to conduct empirical research on the comprehension of process models.

9.2 Outlook

This section covers the potential future functionalities enhancing the applications of this thesis, subdividing it into them respectively. Although progress was made, the focus of this thesis is the inclusion of DyVProMo, the improvement of Usability and Maintainability of the code bases. Therefore, the improvements stated by the initial creators are included as well, marked by an asterisk.

9.2.1 DyVProMo

Saving of process models*

To improve the Applicability of the application, a database to distribute models between multiple users could be added. This removes the need to store the model

files locally and upload them each time the application is used.

Overlay explanation*

The current implementation of overlays is limited to the name of the element. A more sophisticated explanation given in a larger overlay could help novices to gain a faster understanding of the modeling language.

Settings*

Another enhancement could be the provision of a settings dialog to the application in order to only allow dynamic changes to a subset of elements or restrict the number of available tools. The foundation for this is given based on the work of this thesis, as the tools' visibility can be changed programmatically.

Visibility of Swimlanes*

The current version of the application only allows for the highlighting of specific Pools or Lanes. Larger process models may include many of those, making the highlighting feature not sufficient. Therefore, the ability to show only one selected Swimlane could be added.

Traceability of Message Flows

To improve the traceability of Message Flows in large models, a functionality to only show the Lanes that interact with a selected Lane could be added, hiding all others. This may significantly reduce the amount of displayed elements, making the process more comprehensible.

9.2.2 ProMoEE

Access rights for users*

At the moment, users can only see questionnaires they created themselves. Only admins have the right to view all questionnaires. Access rights management to grant users access to specific questionnaires could be added to the application.

Performance on large amounts of answers*

Answers for a single questionnaire are fetched all at once, lowering the performance on large amounts of answers. A paginated approach with dynamic reloading could fix this problem.

Deletion of questionnaires*

Questionnaires cannot be deleted by users at the moment. In production this could lead to a large amount of outdated questionnaires. The ability to delete specific questionnaires or answers could be added.

Alternating questionnaires*

The creation of a functionality for automatically alternating questionnaires could improve the empirical research conducted with support of the application.

Adding color support to modeling

Adding DyVProMo's coloring of elements to the modeler, offering the possibility to research the effect of colored models on model creation, could be added.

Adding color support to Comparator

To compare models with different coloring or no coloring with each other in a more efficient manner, the color support could be added to the Comparator.

Support for the English language

Being only available in German, the application could be enhanced to support the English language. This enables the opportunity to conduct international surveys.

A Sources

This Appendix lists multiple source code files.

Listing A.1: Props of the BpmnViewer with default values

```
1  const BpmnViewer = ({
2    fileData,
3    setFileData,
4    explicitColorLevel,
5    colorLevelCallback,
6    explicitDetailLevel,
7    detailLevelCallback,
8    explicitDetailCheckboxState,
9    detailCheckboxCallback,
10   explicitHighlightedElementIds,
11   highlightedElementIdsCallback,
12   viewerIsEmbedded = false,
13   showDownloadButtons = true,
14   showHighlightBar = true,
15   showColorSlider = true,
16   showToolbar = true,
17 }: BpmnViewerProps) => {
18   [...]
19 }
```

Listing A.2: HighlightBtn Component written in JavaScript

```
1  const HighlightBtn = ({ setShowHighlighter }) => {
2    return (
```

```
3      <button
4        className="btn btn-dark bdv-highlight-btn"
5        onClick={() => {
6          setShowHighlighter(true);
7        }}
8      >
9        Open Highlighter
10     </button>
11   );
12 };
```

Listing A.3: HighlightBtn Component written in TypeScript

```
1 interface HighlightBtnProps {
2   setShowHighlighter: React.Dispatch<React.
3     SetStateAction<boolean>>;
4 }
5 const HighlightBtn = ({ setShowHighlighter }:
6   HighlightBtnProps) => {
7   return (
8     <button
9       className="btn btn-dark bdv-highlight-btn"
10      onClick={() => {
11        setShowHighlighter(true);
12      }}
13    >
14      Open Highlighter
15    </button>
16  );
17 };
```

Listing A.4: CreateUser written as class Component

```
1 interface Params {
2   token: string;
```

```
3 }
4
5 interface Props {
6     requestLoading: boolean;
7     requestFailedText?: string;
8 }
9
10 interface Actions {
11     handleReturnToUserOverview: () => void;
12     handleCreateUser: (token: string, userName: string,
13         password: string) => void;
14 }
15
16 interface State {
17     userName: string;
18     password: string;
19     inputErrorText?: string;
20     error: boolean;
21     showPasswort: boolean;
22 }
23
24 class CreateUserComponent extends React.Component<
25     Params & Props & Actions,
26     State
27 > {
28     constructor(props: Params & Props & Actions) {
29         super(props);
30         this.state = {
31             userName: "",
32             password: "",
33             error: false,
34             showPasswort: false,
35         };
36     }
37 }
```

```
36
37 public componentDidUpdate(prevProps, prevState,
    snapshot) {
38     if (this.props.requestFailedText !== prevProps.
        requestFailedText) {
39         this.setState({ error: true });
40     }
41 }
42
43 public render() {
44     const { requestLoading, requestFailedText,
        handleReturnToUserOverview } =
45         this.props;
46     const { error, showPasswort } = this.state;
47
48     return (
49         <>
50             [...]
51         </>
52     );
53 }
54
55 [...]
56 }
57
58 function mapStateToProps(state: AppState): Props {
59     return {
60         requestLoading: state.intern.requestLoading,
61         requestFailedText: state.intern.requestFailedError,
62     };
63 }
64
65 function mapDispatchToActions(dispatch: AppDispatch):
    Actions {
```

```
66   return {
67     handleReturnToUserOverview: () => dispatch(
        enterUserManagement()),
68     handleCreateUser: (token, userName, password) =>
69       dispatch(createUser(token, userName, password)),
70   };
71 }
72
73 export const CreateUser = connect(
74   mapStateToProps,
75   mapDispatchToActions
76 )(CreateUserComponent);
```

Listing A.5: CreateUser written as functional Component

```
1 interface CreateUserProps {
2   token: string;
3 }
4
5 export const CreateUser = (props: CreateUserProps): JSX.
  Element => {
6   const requestLoading = useSelector(
7     (state: AppState) => state.intern.requestLoading
8   );
9   const requestFailedText = useSelector(
10    (state: AppState) => state.intern.requestFailedError
11  );
12   const dispatch = useDispatch<AppDispatch>();
13
14   const [userName, setUserName] = useState("");
15   const [password, setPassword] = useState("");
16   const [inputErrorText, setInputErrorText] = useState("
    ");
17   const [error, setError] = useState(false);
18   const [showPasswort, setShowPasswort] = useState(false)
```

```
    );  
19  
20    const prevRequestFailedText = useRef<string |  
      undefined>(undefined);  
21  
22    useEffect(() => {  
23      if (requestFailedText !== prevRequestFailedText.  
        current) {  
24        setError(true);  
25      }  
26      prevRequestFailedText.current = requestFailedText;  
27    });  
28  
29    [...]  
30  
31    return (  
32      <>  
33        [...]  
34      </>  
35    );  
36  };
```

Bibliography

- [1] Abramov, Dan. *Getting Started with Redux / Redux*. 2022. URL: <https://redux.js.org/introduction/getting-started> (visited on 03/06/2022).
- [2] Abramov, Dan. *React Redux*. 2022. URL: <https://react-redux.js.org/> (visited on 03/06/2022).
- [3] Becker, Jörg, Rosemann, Michael, and Uthmann, Christoph von. "Guidelines of business process modeling." In: *Business process management*. Ed. by van der Aalst, W., Desel, J., and Oberweis, A. Lecture Notes in Computer Science. Springer, 2000, pp. 30–49.
- [4] Bootstrap team. *Bootstrap · The most popular HTML, CSS, and JS library in the world*. 2022. URL: <https://getbootstrap.com/> (visited on 03/06/2022).
- [5] Camunda Services GmbH. *BPMN editor | demo.bpmn.io*. 2022. URL: <https://demo.bpmn.io/> (visited on 03/07/2022).
- [6] Camunda Services GmbH. *bpmn-js walkthrough*. 2022. URL: <https://bpmn.io/toolkit/bpmn-js/walkthrough> (visited on 03/05/2022).
- [7] Choi, David. *Full-Stack React, TypeScript, and Node*. 1st ed. Boston, MA: Packt Publishing, 2020. ISBN: 978-1-83921-993-1.
- [8] Cohen-Or, Daniel et al. "Color harmonization." In: *ACM Trans. Graph.* 25.3 (2006), pp. 624–630. ISSN: 0730-0301. DOI: 10.1145/1141911.1141933.
- [9] Dropwizard Team. *Home - Dropwizard*. 2022. URL: <https://www.dropwizard.io/en/latest/> (visited on 03/06/2022).
- [10] Dubey, Sanjay Kumar and Rana, Ajay. "Analytical roadmap to usability definitions and decompositions." In: *International Journal of Engineering Science and Technology* 2.9 (2010), pp. 4723–4729.

- [11] Dumas, Marlon et al. *Fundamentals of business process management*. Vol. 1. Springer, 2013. DOI: 10.1007/978-3-642-33143-5.
- [12] Elrom, Elad. *React and Libraries: Your Complete Guide to the React Ecosystem*. 1st ed. New York: Apress, 2021. ISBN: 978-1-4842-6696-0.
- [13] Erol, Selim. "Coloring support for process diagrams: a review of color theory and a prototypical implementation." In: *Vienna University of Economics and Business* (2015).
- [14] Facebook, Inc. *Getting Started | Create React App*. 2022. URL: <https://create-react-app.dev/docs/getting-started> (visited on 03/06/2022).
- [15] Facebook, Inc. *Introducing Hooks – React*. 2022. URL: <https://reactjs.org/docs/hooks-intro.html> (visited on 03/06/2022).
- [16] Facebook, Inc. *React - A JavaScript library for building user interfaces*. 2022. URL: <https://reactjs.org/> (visited on 03/06/2022).
- [17] Figl, Kathrin, Recker, Jan, and Mendling, Jan. "A Study on the Effects of Routing Symbol Design on Process Model Comprehension." In: *Decision Support Systems* 54.2 (2013), pp. 1104–1118. ISSN: 0167-9236. DOI: 10.1016/j.dss.2012.10.037. URL: <https://www.sciencedirect.com/science/article/pii/S0167923612003119>.
- [18] Fowler, Martin. *Refactoring: Improving the Design of Existing Code*. 2nd ed. Boston: Addison-Wesley, 2019. ISBN: 978-0-13-475768-1.
- [19] Gallik, Florian. "Dynamic Visualization of Additional Information in Process Models." Master's Thesis. Ulm University, 2021.
- [20] Genon, Nicolas, Heymans, Patrick, and Amyot, Daniel. "Analysing the cognitive effectiveness of the BPMN 2.0 visual notation." In: *Proceedings of the Third international conference on Software language engineering*. Ed. by Malloy, Brian, Staab, Steffen, and van den Brand, Mark. Lecture Notes in Computer Science. Springer, 2010, pp. 377–396.

- [21] Green, T. R. G. and Petre, M. "Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework." In: *Journal of Visual Languages & Computing* 7.2 (1996), pp. 131–174. ISSN: 1045-926X. DOI: 10.1006/jvlc.1996.0009. URL: <https://www.sciencedirect.com/science/article/pii/S1045926X96900099>.
- [22] Griggs, Bethany. *Node Cookbook*. 4th ed. Boston: Packt Publishing, 2020. ISBN: 978-1-83855-875-8.
- [23] Gutermuth, Jonas. "Konzeption und Implementierung einer webbasierten Anwendung zur Durchführung von Modellierungsexperimenten." Bachelor's Thesis. Ulm University, 2016.
- [24] Healey, C. and Enns, J. "Attention and Visual Memory in Visualization and Computer Graphics." In: *IEEE Transactions on Visualization and Computer Graphics* 18.7 (2012), pp. 1170–1188. DOI: 10.1109/TVCG.2011.127.
- [25] Jin, Fang and Kale, Sagar. *Designing React Hooks the Right Way*. 1st ed. Boston, MA: Packt Publishing, 2022. ISBN: 978-1-80323-595-0.
- [26] Kim, Miryung, Zimmermann, Thomas, and Nagappan, Nachiappan. "A field study of refactoring challenges and benefits." In: *Proceedings of the 20th International Symposium on the Foundations of Software Engineering*. Cary, North Carolina: Association for Computing Machinery, 2012, Article 50. ISBN: 9781450316149. DOI: 10.1145/2393596.2393655.
- [27] Kolb, Jens, Kammerer, Klaus, and Reichert, Manfred. "Updatable Process Views for Adapting Large Process Models: The proView Demonstrator." In: *Demo Track of the 10th International Conference on Business Process Management*. Lecture Notes in Computer Science. Springer, 2012, pp. 6–11. URL: <http://dbis.eprints.uni-ulm.de/847/>.
- [28] Kolb, Jens and Reichert, Manfred. "A Flexible Approach for Abstracting and Personalizing Large Business Process Models." In: *Applied Computing Review* 13.1 (2013), pp. 6–17. URL: <http://dbis.eprints.uni-ulm.de/914/>.
- [29] Krause, Jörg. *Introducing Bootstrap 4: create powerful web applications using Bootstrap 4.5*. 2nd ed. Berkeley, CA: Apress, 2020. ISBN: 978-1-4842-6203-0.

- [30] Kreßmann, Fabian. “Konzeption und Entwicklung einer Web-Plattform für die Definition, Durchführung und Auswertung von Studien im Kontext von Business Process Management.” Master’s Thesis. Ulm University, 2019.
- [31] Kummer, Tyge-F., Recker, Jan, and Mendling, Jan. “Enhancing Understandability of Process Models through Cultural-dependent Color Adjustments.” In: *Decision Support Systems* 87 (2016), pp. 1–12. ISSN: 0167-9236. DOI: 10.1016/j.dss.2016.04.004. URL: <https://www.sciencedirect.com/science/article/pii/S0167923616300574>.
- [32] La Rosa, M. et al. “Managing Process Model Complexity via Concrete Syntax Modifications.” In: *IEEE Transactions on Industrial Informatics* 7.2 (2011), pp. 255–265. DOI: 10.1109/TII.2011.2124467.
- [33] Leopold, H., Mendling, J., and Günther, O. “Learning from Quality Issues of BPMN Models from Industry.” In: *IEEE Software* 33.4 (2016), pp. 26–33. ISSN: 1937-4194. DOI: 10.1109/MS.2015.81.
- [34] MacDonald, L. W. “Using color effectively in computer graphics.” In: *IEEE Computer Graphics and Applications* 19.4 (1999), pp. 20–35. DOI: 10.1109/38.773961.
- [35] Matsuda, Y. *Color Design*. Asakura Shoten, 1995.
- [36] Mendling, Jan, Reijers, Hajo A., and Cardoso, Jorge. “What makes process models understandable?” In: *International Conference on Business Process Management*. Ed. by Alonso, G., Dadam, P., and Rosemann, M. Lecture Notes in Computer Science. Springer, 2007, pp. 48–63.
- [37] Mens, T. and Tourwe, T. “A survey of software refactoring.” In: *IEEE Transactions on Software Engineering* 30.2 (2004), pp. 126–139. DOI: 10.1109/TSE.2004.1265817.
- [38] Meyer, Eric A. and Weyl, Estelle. *CSS: The Definitive Guide: Visual Presentation for the Web*. 4th ed. Sebastopol, CA: O’Reilly Media, 2018.
- [39] Microsoft. *TypeScript: Documentation - The Basics*. 2022. URL: <https://www.typescriptlang.org/docs/handbook/2/basic-types.html> (visited on 03/06/2022).

- [40] Moody, D. "The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering." In: *IEEE Transactions on Software Engineering* 35.6 (2009), pp. 756–779. DOI: 10.1109/TSE.2009.67.
- [41] Mozilla Foundation. *Classes - JavaScript / MDN*. 2022. URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes> (visited on 03/06/2022).
- [42] Natschläger, Christine. "Deontic BPMN." In: *International Conference on Database and Expert Systems Applications*. Ed. by Hameurlain, A. et al. Lecture Notes in Computer Science. Springer, 2011, pp. 264–278.
- [43] Nielsen, Jakob. *Usability Engineering*. Morgan Kaufmann, 1994. ISBN: 0125184069.
- [44] OMG BPMN Model Interchange Working Group. *BPMN in Color Specification*. 2014. URL: <https://github.com/bpmn-miwb/bpmn-in-color/blob/master/BPMN%20in%20COLOR.pdf> (visited on 03/04/2022).
- [45] OMG BPMN Model Interchange Working Group. *The BPMN Model Interchange Working Group*. 2022. URL: <https://www.omgwiki.org/bpmn-miwb/doku.php> (visited on 03/04/2022).
- [46] OMG. *BPMN 2.0 by Example*. 2010. URL: <https://www.omg.org/cgi-bin/doc?dte/10-06-02.pdf> (visited on 03/05/2022).
- [47] OMG. *Business Process Model and Notation (BPMN)*. 2014. URL: <https://www.omg.org/spec/BPMN/2.0>.
- [48] Pinggera, Jakob, Zugal, Stefan, and Weber, Barbara. "Investigating the Process of Process Modeling with Cheetah Experimental Platform." In: *Proceedings of the 1st Empirical Research in Process-Oriented Information Systems*. Ed. by Mutschler, B. et al. CEUR, 2010, pp. 13–18.
- [49] Rappin, Noel. *Modern CSS with Tailwind*. 1st ed. Boston, MA: Pragmatic Bookshelf, 2021.
- [50] Recker, Jan. "Empirical Investigation of the Usefulness of Gateway Constructs in Process Models." In: *European Journal of Information Systems* 22.6 (2013), pp. 673–689. ISSN: 0960-085X. DOI: 10.1057/ejis.2012.50.

- [51] Recker, Jan et al. "Business Process Modeling-A Comparative Analysis." In: *Journal of the association for information systems* 10.4 (2009), pp. 333–363. ISSN: 1536-9323. DOI: 10.17705/1jais.00193.
- [52] Reijers, H. A. et al. "Syntax highlighting in business process models." In: *Decision Support Systems* 51.3 (2011), pp. 339–349. ISSN: 0167-9236. DOI: 10.1016/j.dss.2010.12.013. URL: <https://www.sciencedirect.com/science/article/pii/S0167923611000042>.
- [53] Scherle, Steffen. "Konzeption und Evaluierung einer domänenspezifischen Modellierungsumgebung für prozessorientierte Fragebögen." Diploma Thesis. Ulm University, 2014.
- [54] Schrepfer, Matthias et al. "The Impact of Secondary Notation on Process Model Understanding." In: *IFIP Working Conference on The Practice of Enterprise Modeling*. Ed. by Persson, A. and Stirna, J. Lecture Notes in Business Information Processing. Springer, 2009, pp. 161–175.
- [55] Stark, J., Braun, R., and Esswein, W. "Perceptually Discriminating Chunks in Business Process Models." In: *Proceedings of the 18th Conference on Business Informatics*. 2016, pp. 84–93. ISBN: 2378-1971. DOI: 10.1109/CBI.2016.18.
- [56] Stark, Jeannette. "Perceptual Discriminability in Conceptual Modeling." In: *Proceedings of the 6th Enterprise Engineering Working Conference*. Ed. by Aveiro, D., Pergl, R., and Gouveia, D. Lecture Notes in Business Information Processing. Springer, 2016, pp. 103–117.
- [57] Stark, Jeannette and Esswein, Werner. "Using Secondary Notation to Improve the Cognitive Effectiveness of BPMN-Models." In: *Proceedings of the 25th European Conference on Information Systems*. Ed. by Ramos, Isabel, Tuunainen, Virpi, and Krcmar, Helmut. Association for Information Systems, 2017. ISBN: 978-989-20-7655-3. URL: https://aisel.aisnet.org/ecis2017_rp/35.
- [58] Stark, Jeannette, Esswein, Werner, and Braun, Richard. "Systemizing Colour for Conceptual Modeling." In: *Proceedings of the 13th International Conference on Wirtschaftsinformatik*. Ed. by Leimeister, J. M. and Brenner, W. 2017, pp. 256–270.

- [59] Stein Dani, Vinicius, Dal Sasso Freitas, Carla Maria, and Thom, Lucinéia Heloisa. "Ten years of visualization of business process models: A systematic literature review." In: *Computer Standards & Interfaces* 66 (2019), p. 103347. ISSN: 0920-5489. DOI: 10.1016/j.csi.2019.04.006. URL: <https://www.sciencedirect.com/science/article/pii/S0920548918303295>.
- [60] *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML*. 2022. URL: <https://tailwindcss.com/> (visited on 03/06/2022).
- [61] Thomas, Jeremy. *Bulma: Free, open source, and modern CSS framework based on Flexbox*. 2016. URL: <https://bulma.io/> (visited on 03/06/2022).
- [62] Tobias Paul Bleisch. "An Empirical Study of CSS Code Smells in Web Frameworks." Master's Thesis. California Polytechnic State University, 2018. DOI: 10.15368/theses.2018.105.
- [63] Tokumaru, M., Muranaka, N., and Imanishi, S. "Color design support system considering color harmony." In: *Proceedings of the IEEE International Conference on Fuzzy Systems*. 2002, pp. 378–383. DOI: 10.1109/FUZZ.2002.1005020.
- [64] Ulm University. *QuestionSys - A Generic and Flexible Questionnaire System Enabling Process-Driven Mobile Data Collection*. 2013. URL: <https://www.uni-ulm.de/in/iui-dbis/forschung/laufende-projekte/questionsys> (visited on 03/02/2022).
- [65] Wong, Wucius. *Principles of Color Design: Designing with Electronic Color*. Van Nostrand Reinhold, 1997.
- [66] npm, Inc. *About npm*. 2022. URL: <https://docs.npmjs.com/about-npm> (visited on 03/05/2022).
- [67] npm, Inc. *package.json | npm Docs*. 2022. URL: <https://docs.npmjs.com/cli/v8/configuring-npm> (visited on 03/05/2022).
- [68] van der Aalst, Wil M. P. et al. "Business Process Management: A Comprehensive Survey." In: *ISRN Software Engineering* (2013). DOI: 10.1155/2013/507984.

Name: Florian Loth

Matrikelnummer: 994158

Erklärung

Ich erkläre, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den 08.03.2022... 

Florian Loth