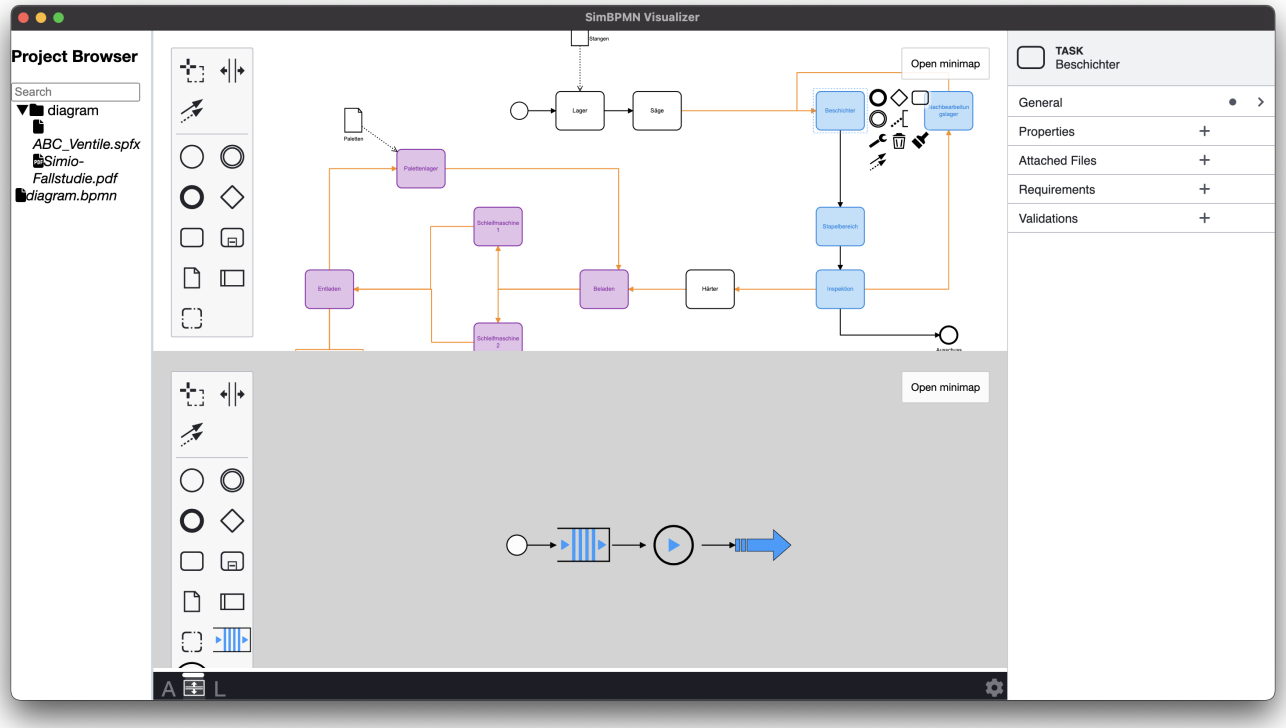
OST
Ostschweizer
Fachhochschule

Campus Rapperswil-Jona
8640 Rapperswil

Bachelorarbeit



simBPMN Modellierungssoftware

Frühjahrssemester 2022

Autoren: Michel Mirsayyah
Sven Höpfner

Dozent: Prof. Dr. Andreas Rinkel

Themengebiet: Software-Engineering

Experte: Knut Schmahl

Studiengang: Informatik

Gegenleser: Ivan Bütler

Inhaltsverzeichnis

Bildverzeichnis

Tabellenverzeichnis

0.1 Abstract

Problemstellung

Um Prozesse mit Simulationsbestand in einem System auf einer hohen Abstraktionsebene zu beschreiben, wurde von der ASIM-Fachgruppe "Einsatz formaler Methoden und Vorgehensmodellen zur Simulation" die simBPMN Notation ausgearbeitet. Im Rahmen dieser Arbeit wird ein Spezifikationstool entwickelt, dass die effiziente Nutzung der Spezifikation mit Hilfe der simBPMN unterstützt. Neben der syntaktischen Korrektheit der Spezifikation werden weitere nicht formale Funktionen definiert und eingebaut, z.B. Definition von Anforderungen, Parameter zur Validierung und Verifikation und das Einbinden zusätzlicher Dokumente. Dieses ermöglicht es, Prozesse genauer zu spezifizieren und den Austausch zwischen verschiedenen Benutzern zu verbessern. Zusätzlich ist es wichtig, dass die Applikation auf den Betriebssystemen Windows, Linux und MacOS verwendet werden kann.

Methode / Vorgehen

Die Software wird mithilfe der agilen Methode Scrum+ entwickelt. Wöchentlich wird der Stand der Arbeit mit dem verantwortlichen Betreuer der Ostschweizer Fachhochschule besprochen. Da es sich um eine erste Version handelt wird an den Besprechungen Feedback entgegengenommen besprochen und in den Arbeitsplan mit eingearbeitet.

Ergebnisse

Die erstellte Lösung ermöglicht es zu jeden einzelnen Element in einem Prozess, eine Prozesslogik zu spezifizieren. Jedes Element kann ausserdem mit einem Anhang, unabhängig von der Applikation, genauer spezifiziert werden. Durch die zusätzlich implementierten Export- und Importfunktionen eines Projektes können die erstellten Spezifikationen ausgetauscht werden.

Kapitel 1

Aufgabenstellung

1.1 Einleitung

In der Geschäftswelt treffen immer wieder unterschiedliche Welten aufeinander. Dies wird unter anderem bei der Beschreibung von Simulationsmodellen deutlich. Während Geschäftskunden sich dafür lieber einer umgangssprachlichen Sprache bedienen, bevorzugen Fachexperten eine formale oder semiformale Sprache, wie zum Beispiel Systems Modeling Language (SysML). Um die Akzeptanz formaler Sprachen zu erhöhen, wurde die vereinfachte BPMN 2.0 eingeführt. Diese wird vor allem in der Wirtschaftsinformatik und im Prozessmanagement eingesetzt, wo vermehrt Berührungspunkte entstehen.
Da es der BPMN Standard allerdings nicht ermöglicht ein Prozessverhalten zu beschreiben, wurde in der ASIM-Fachgruppe "Einsatz formaler Methoden und Vorgehensmodellen zur Simulation", die simBPMN Notation als Ergänzung zur BPMN erarbeitet. Diese ermöglicht es Prozesse mit Simulationsbestand auf einem hohen Abstraktionsniveau zu beschreiben.

1.2 Auftrag

In einer Vorarbeit wurden verschiedenste Editoren und Möglichkeiten analysiert, mit welchen ein Modell mittels der simBPMN Notation beschrieben werden kann. Das Ergebnis dieser Analyse hat ergeben, dass sich das Javascript Toolkit BPMN IO am besten für eine solche Umsetzung eignet.
Nach der vorangegangenen Analyse wird nun in dieser Folgearbeit eine Applikation entwickelt, mithilfe welcher die simBPMN-Notation eingesetzt werden kann. Die Applikation wird dafür das BPMN IO Toolkit einsetzen und um weitere Funktionalitäten erweitert.

Kapitel 2

Management Summary

2.1 Kontext / Problemstellung

In der Geschäftswelt treffen immer wieder unterschiedliche Welten aufeinander. Dies wird unter anderem bei der Beschreibung von Simulationsmodellen deutlich. Während Geschäftskunden sich dafür lieber einer umgangssprachlichen Sprache bedienen, bevorzugen Fachexperten eine formale oder semiformale Sprache, wie zum Beispiel die Systems Modeling Language (SysML). Um die Akzeptanz formaler Sprachen zu erhöhen, wurde die vereinfachte BPMN 2.0 eingeführt. Diese wird vor allem in der Wirtschaftsinformatik und im Prozessmanagement eingesetzt, wo vermehrt Berührungspunkte entstehen.

Da es der BPMN Standard allerdings nicht ermöglicht ein Prozessverhalten zu beschreiben, wurde in der ASIM-Fachgruppe "Einsatz formaler Methoden und Vorgehensmodellen zur Simulation", die simBPMN Notation als Ergänzung zur BPMN erarbeitet. Diese ermöglicht es Prozesse mit Simulationsbestand auf einem hohen Abstraktionsniveau zu beschreiben.

Für die simBPMN existiert allerdings derzeit kein Spezifikationstool, welches die Notation unterstützt. In einer Vorarbeit wurde analysiert, wie man ein solches Tool umsetzen kann. Als Grundlage für die Entwicklung wird das BPMN IO Framework verwendet.

2.2 Ziel

Im Rahmen dieser Bachelorarbeit wird eine erste digitale Möglichkeit erschaffen, mithilfe der simBPMN Notation Prozesse zu spezifizieren. Dies ermöglicht es Simulationsaspekte mit in die Spezifikation von Prozessen einzubringen. Durch verschiedene Features der Applikation soll die Prozessspezifikation zusätzlich einfacher und verständlicher gemacht werden. Ausserdem ist es notwendig, dass das Spezifikationstool auf einem Rechner mit Windows, MacOS oder Linux ohne Umstände ausgeführt werden kann.

2.3 Rahmenbedingungen

Da das Spezifikationstool auf das BPMNIO Framework aufbaut, wird die Entwicklung mit den Programmiersprachen Javascript und Typescript durchgeführt. Um eine Cross-Plattform Anwendung zu ermöglichen, basiert die Applikation auf dem Electron Framework. Die Applikation muss nicht installiert werden, um sie auf einem der unterstützten Rechnern laufen lassen zu können.

2.4 Vorgehen

Das Vorgehen im Projekt ist ablauforientiert und wird durch Meilensteine in zeitliche Abschnitte unterteilt. Als Vorgehensmodell kommt eine Mischung aus Scrum und dem Unified Process zum Einsatz. Aus dem Unified Process werden die drei Phasen Elaboration, Construction und Transition übernommen. Innerhalb der Phasen wird in wochenbasierten Iterationen gearbeitet. In der ersten Phase wird ein Projektplan erstellt, sowie die Use Cases und User Storys erarbeitet. In der Konstruktionsphase werden schliesslich die so definierten Anforderungen umgesetzt und durch stetiges Feedback erweitert. In der finalen Transitionsphase wird die Anwendung für eine Distribution vorbereitet und an den Betreuer übergeben.

2.5 Resultate

Die vom Betreuer definierten Anforderungen konnten in einer ersten Version umgesetzt werden. Die Applikation ermöglicht es mithilfe der in der simBPMN Notation definierten Elemente Prozesse zu spezifizieren. Mithilfe einer eingebauten Splitview kann man für jedes Element eine Prozesslogik spezifizieren. Jedes Element bietet zudem die Möglichkeit, es noch detailierter zu beschreiben. Dies ist mit wörtlichen Beschreibungen, Anhängen oder Eigenschaften möglich. Alle Dateien der Applikation werden in einem Workspace gespeichert. Die Basisdatei in welcher der Prozess modelliert wird, wird in einem XML-Format abgespeichert. Die Anhänge der Datei werden in einem Unterordner kopiert, welcher den entsprechenden Namen der Basisdatei besitzt. Dies ermöglicht es, die Anhänge ihren Basisdateien zuordnen zu können. Diese Struktur bietet gleichzeitig auch den Grundstein die Dateien, in einem Komprimierten Zip-Format exportieren und importieren zu können. Die Applikation wurde mithilfe von Electron geschrieben. Dies ermöglicht es die Applikation die bekanntesten Betriebsysteme bereitzustellen. Zusätzlich wird die Mehrsprachigkeit unterstützt.

2.6 Ausblick

Die erste Version wird von Prof. Dr. Andreas Rinkel sowie der ASIM Gruppe getestet und weiter verteilt. Durch die effektive Benutzung von verschiedenen Parteien lassen sich Probleme und verbesserungswürdige Stellen feststellen. Die Applikation ist noch keineswegs ein fertiges Produkt und bietet viel Platz für weitere Verbesserungen, welche in dieser Arbeit, aus zeitlichen Gründen, nicht umgesetzt wurden konnten.
Durch die Umsetzung der Basisfunktionalitäten für eine Logikspezifikation, bietet die entwickelte Applikation jedoch bereits eine gute Basis für weitere Funktionalitäten. Durch den speziellen modularen Aufbau des BPMN IO Frameworks ist es möglich weitere, eigen entwickelte, Module an die Applikation anzubinden. Diese können beinhalten, dass man einen Prozess validieren oder sogar durchlaufen kann.

Kapitel 3

Technischer Bericht

3.1 Einleitung

3.1.1 Form der Arbeit

Teile der Arbeit werden dem betreuenden Dozenten bereits während des Projekts durch Vorabveröffentlichungen zur Verfügung gestellt. Diese werden mit dem Abschluss der im Projektplan definierten Meilensteine veröffentlicht.

3.1.2 Ausgangslage & Problemstellung

Der BPMN 2.0 Standard ist eine grafische Spezifikationssprache, die es erlaubt, Arbeitsabläufe und ihre Prozesse zu modelieren. Die BPMN Notation erlaubt es jedoch nicht zwischen der Architektur und der Logik des Ablaufes zu unterscheiden. Ferner kennt die BPMN zur Zeit nur sehr eingeschränkte Möglichkeiten zur Modellierung von Zeit und Ressourceneinsätzen.
Mithilfe der neu entworfenden simBPMN Notation diese Mängel behoben, ohne dafür die ursprüngliche Einfachheit der BPMN einzuschränken. Zum einen wurde die Anzahl der Elemente auf ein Minimum beschränkt und zum anderen wurde das Konzept des Entity-Flows[1] eingefügt.
Da die simBPMN eine neu spezifizierte Notation ist, existieren derzeit keine Werkzeuge, welche diese unterstützen. Somit besteht derzeit keine Möglichkeit, einen Prozess mithilfe der Notation technisch zu Modellieren.

3.1.3 Ziele

Das Ziel der Arbeit besteht darin, eine Applikation bereitzustellen, mit dessen Hilfe ein System zur Simulation durch die simBPMN beschrieben werden kann. Weitere unterstützende Funktionen, welche die Applikation beinhalten soll, werden in der Anforderungsdefinition spezifiert

[1] Der Entity-Flow beschreibt den Fluss einer Entität durch einen Prozess. Durch dieses Konzept kann die Archtitektur des Prozesses dargestellt werden.

3.2 Beschreibung BPMN

Die Business Prozess Model and Notation Spezifikation, kurz BPMN, bietet eine grafische Notation, um Geschäftsprozesse in einem Diagramm abzubilden. Das Ziel dieser Notation ist es, eine Darstellung zu bieten, die einerseits intuitiv verständlich ist und mit wenigen Sprachelementen auskommt, andererseits soll es nicht zu einem konzeptuellen Bruch kommen, wenn weitergehende Formalisierungen gefordert sind.
Die einfachste Form der BPMN Notation, welche als Basis für simBPMN verwendet wurde, beinhaltet folgende Elemente.[2]

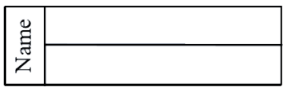
Element	Name	Beschreibung
	Aufgabe	Ein Aufgabe ist eine Arbeitseinheit und kann nicht weiter unterteilt werden.
	Teilprozess	Ein Teilprozess kapselt mehrere Aufgaben oder Teilprozesse zu einer logisch zusammenhängenden Einheit.
	Ereignis	Ein Ereignis beschreibt etwas, was zu einem bestimmten Zeitpunkt im Prozess passiert und kann dabei Tokens erzeugen.
	Gateway	Mit Gateways werden Fallunterscheidungen dargestellt.
	Sequenzfluss	Durch den Sequenzfluss kann man die Ausführungsreihenfolge des Geschäftsprozesses definieren.
	Nachrichtenfluss	Eingehende Nachrichten / Materialien in den Geschäftsprozesses können durch den Nachrichtenfluss dargestellt werden.
	Assoziation	Durch eine Assoziation können zusammenhängende Objekte verbunden werden.
	Datenobjekt	Ein Datenobjekt repräsentiert ein Artefakt, welches durch den Geschäftsprozess bearbeitet wird.
	Textanmerkung	Eine Textanmerkung kann einem Element eines Geschäftsprozesses zugeordnet werden.
	Gruppierung	Eine Gruppierung ist ein Hilfsmittel, um Elemente eines Geschäftsprozesses visuell zusammenzufassen
	Pool	Der Pool umfasst einen einzigen Geschäftsprozess und besteht aus einem oder mehreren Lanes.
	Lanes	Die Lanes repräsentieren die unterschiedlichen Teilnehmer des Geschäftsprozesses. Lanes beinhalten entsprechende Aufgaben und Ereignisse des Teilnehmers.

Tabelle 3.1: BPMN Standardelemente

[2] Die Icons basieren auf der Darstellung von BPMN IO und sind öffentlich zugänglich unter https://github.com/bpmn-io/bpmn-font

3.2.1 Ereignisse

Ereignisse symbolisieren das Auftreten und Auslösen von Geschäftsvorfällen innerhalb eines Geschäftsprozesses. Ereignisse können entweder "geworfen" oder "gefangen" werden und besitzen immer einen der folgenden drei Zustände:

Element	Name	Beschreibung
	Start	Beschreibt den Anfang eines Prozesses
	Zwischen	Ein Ereignis, welches sich im Zwischenzustand befindet und während des Prozesses jederzeit geworfen oder gefangen werden kann.
	Ende	Beschreibt das Ende eines Prozesses

Tabelle 3.2: BPMN Ereignisse

Die Ereignisse können zusätzlich verschieden typisiert werden. In der simBPMN wird sich auf die folgenden Typen beschränkt:

Typ	Name	Beschreibung
	Timer	Das Ereignis wird zu einem festgelegten Zeitpunkt oder nach einem Zeitintervall ausgelöst.
	Fehler	Wenn die Möglichkeit besteht, dass Fehler im Prozess auftreten können, ist es möglich diese mit dem Fehler Ereignis darzustellen. Der Benutzer muss zusätzlich definieren um was für ein Fehler es sich handelt, da dies nicht aus dem Ereignis ersichtlich ist.
	Terminierung	Durch die Terminierung werden alle noch laufenden Prozesse abgebrochen. Es wird eine sofortige Beendigung ausgelöst.

Tabelle 3.3: BPMN Ereignis Typen

Kompositionen

Anstatt Ereignisse nach einer Aufgabe in den Fluss einzubinden, können sie auch an einen Prozess angebunden werde. Im Fall des Autretens, wird dann der laufende Prozess unterbrochen und der angebundene Zweig ausgeführt. Im folgenden Beispiel terminiert das Fehler Ereignis die Aufgabe und startet die Nachfolgende.

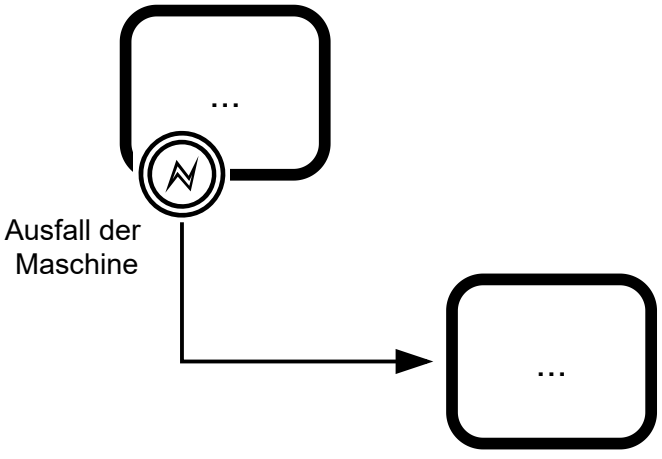


Bild 3.1: Ereignisse als Komposition

3.2.2 Gateways

Gateways beschreiben Verzweigungen im Prozessverlauf und werden in zwei Arten unterschieden. Es gibt datenbasierte Gateways, welche den Zustand der im Prozess vorkommenden Daten und deren Konstallation auswertet und ereignisbasierte Gateways, welche von den Geschäftsvorfällen abhängig sind. Es werden folgende Gateways in der Arbeit berücksichtigt:

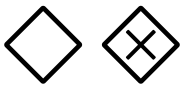
Element	Name	Beschreibung
	Exklusives Gateway (XOR)	Aus den verschiedenen Optionen wird nur ein Datenstrom ausgewählt. Dieses Gateway wird auch zum Zusammenführen der Datenströme genutzt.
	Inklusives Gateway (OR)	Aus den verschiedenen Optionen werden alle Datenströme ausgewählt, welche die Bedingung erfüllen. So kann der Datenstrom in eine, mehrere oder alle verfügbaren Richtungen aufgeteilt werden.
	Paralleles Gateway (AND)	Der Datenstrom wird in alle Richtungen aufgeteilt.
	Ereignisbasiertes Gateway	Das ereignisbasierte Gateway wartet auf ein Ereignis und führt je nach dem, welches Ereignis gefangen wurde, einen anderen Vorgang aus.
	Exklusives ereignisbasiertes Gateway	Das exklusive ereignisbasierte Gateway wird verwendet, um für jedes mögliche Ereignis einen Prozess zu starten. Wenn ein neues Ereignis eintretet, wird ein neuer unabhängiger Prozess gestartet.
	Paralleles ereignisbasiertes Gateway	Um das parallele ereignisbasierte Gateway auslösen zu können, müssen alle definierten Ereignisse eingetreten sein, bevor der Prozess gestartet werden kann.

Tabelle 3.4: BPMN Gateways

3.2.3 Beispiele

Allgemeiner Prozess

Ein einfacher Geschäftsprozess kann mit den vorher beschriebenen Element umgesetzt werden. Ein Beispiel dazu ist nachvollgend abgebildet.

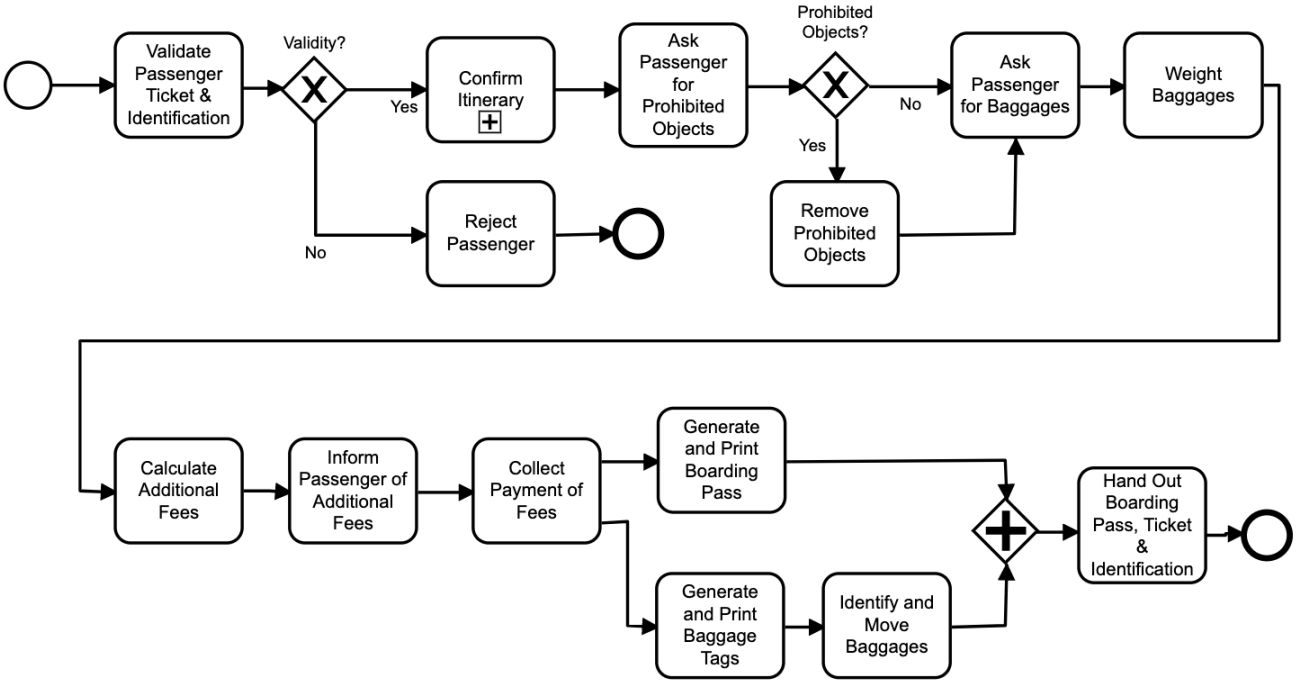


Bild 3.2: Einfacher Prozess in BPMN

Ereignisbasierte Gateways

Die ereignisbasierten Gateways werden jeweils vor den Ereignissen definiert.

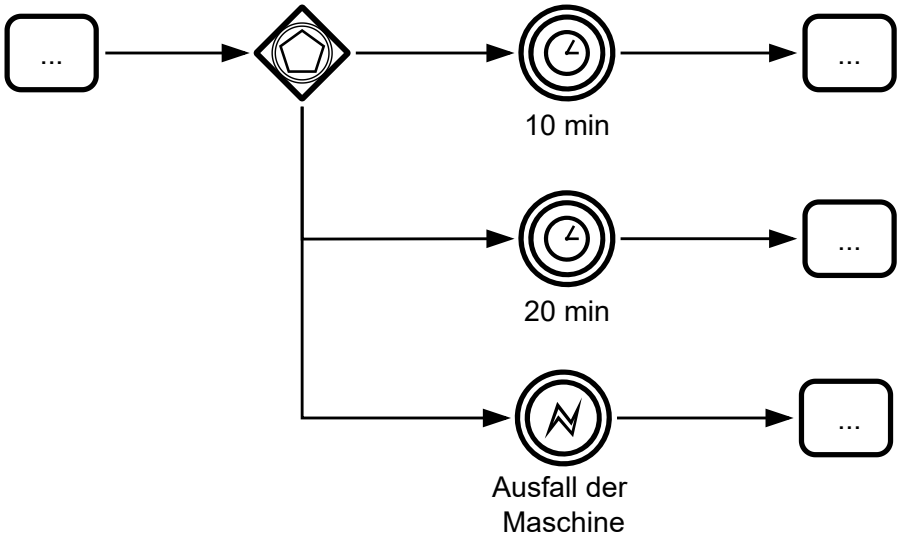


Bild 3.3: Erreignis basiertes Beispiel

3.3 Beschreibung simBPMN

SimBPMN ist eine Erweiterung des BPMN 2.0 Standards. Die simBPMN Notation verwendet nur die im Kapitel Beschreibung BPMN beschriebenen Elemente. Zusätzlich wird die Notation um Elemente erweitert, welche dem Benutzer die Möglichkeit gibt, die System-Architektur eines Prozesses zu beschreiben.

Elementname	Beschreibung
Entität	Eine Entität läuft mithilfe des Sequenzfluss von Teilprozess zu Teilprozess und löst bei der Ankunft das Start Ereignis aus wodurch ein Token erstellt wird.
Token	Ein Token besitzt einen Zustand und fliesst durch einen Teilprozess. Er ändert dabei die Zustände von Entitäten, Ressourcen oder anderen Teilprozessen. Es besitzt die Fähigkeit Ereignisse auszulösen.
Ressourcen	Wird für die Ausführung eines Teilprozesses benötigt

Tabelle 3.5: simBPMN Elemente

Um die folgenden beiden Konzepte verständlicher zu gestalten, werden diese mit einem Beispielprozess dargestellt.

Prozessbeispiel

Die Firma "Ikarus" ist ein Flugzeughersteller. Um Flugzeuge effektiv herstellen zu können wurde ein Fliessbandsystem eingeführt. Flugzeuge werden nun auf diesem Fliessband von Station zu Station transportiert. An jeder Station wird eine Aufgabe am Flugzeug erledigt. Somit wird an der ersten Station das Chassis zusammengesetzt, an der zweiten die Flügel und Ruder mit der Lenkung verknüpft und an der dritten die Innenverkleidung eingebaut.

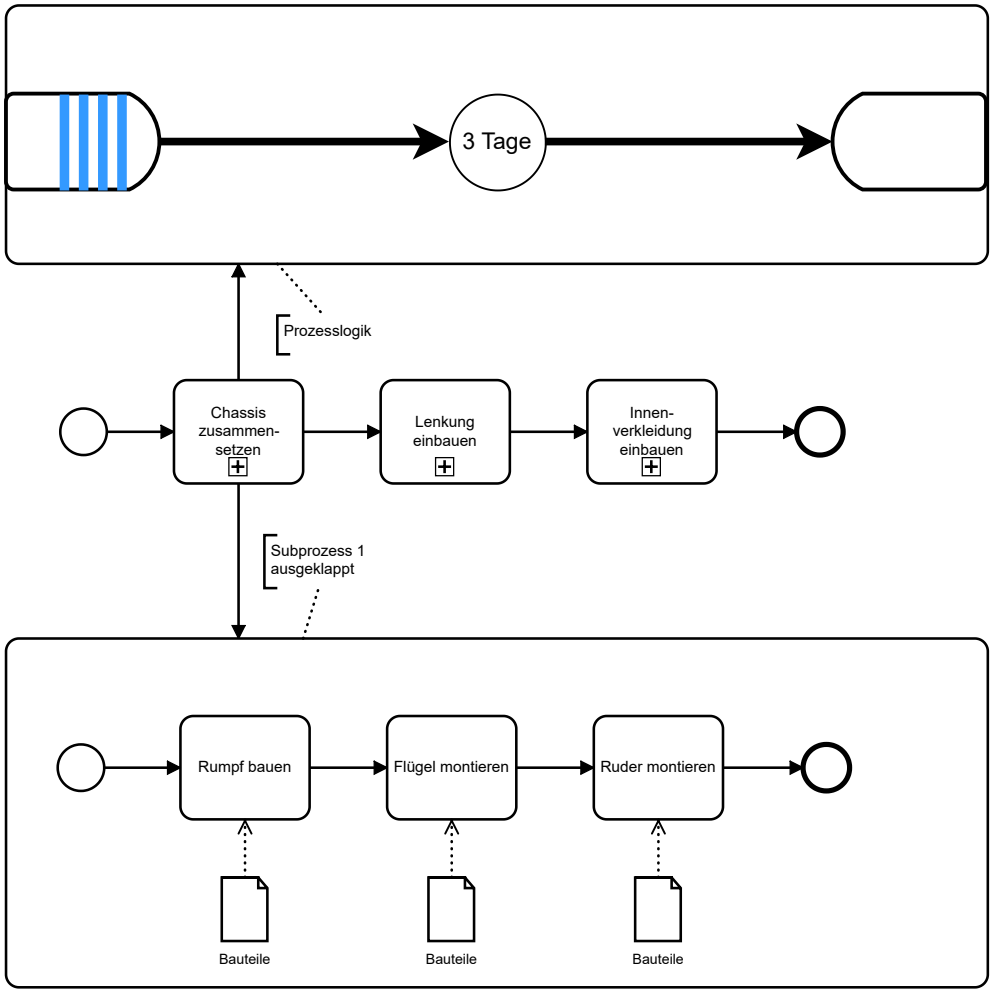


Bild 3.4: Beispielprozess Ikarus

3.3.1 Architektur/Struktur (Entity-Flow)

Im Prozessbeispiel ist das Flugzeug die Entität des Prozesses. Entitäten bestimmen den Fluss des Geschäftprozesses und werden dabei als Input für die verschiedenen Teilprozesse verwendet. Die Teilprozesse verbrauchen ihre Entität, erstellen jedoch dafür eine neue, welche zum nächsten Teilprozess weitergegeben wird. Der Entity-Flow ist mit der BPMN 2.0 sehr gut beschreibbar. Einzig das Entitätsobjekt fehlt und muss erweitert werden. In der Modellierung ist die Entität allerdings eher ein theoretisches Objekt. Sie kann definiert werden um zu signalisieren, dass verschiedene Entitäten den Prozess durchlaufen können. Im Prozessbeispiel könnten wir somit verschiedene Flugzeugmodelle, welche allerdings den gleichen Prozess durchlaufen mit verschiedenen Entitäten abbilden.

3.3.2 Prozesslogik (Token-Flow)

Die Prozesslogik beschreibt wie ein einzelner Teilprozess abgearbeitet wird. Der Ablauf des Teilprozesses wird mithilfe von Tokens gesteuert. In unserem Beispiel haben wir den Teilprozess, dass das Chassis zusammengesetzt werden muss. Mithilfe des Tokens, welcher wie die Entität ein theoretisches Objekt ist, kann man nun durch den Prozess durchgehen und diesen mit der BPMN 2.0 definieren. Die Prozesslogik beschreibt allerdings nun auch wie mit den Objekten, welche über das Fliessband ankommen umgegangen wird. So befinden sich auf dem Teilprozess die Eigenschaften des Eingangspuffer / der Warteschlange, die allgemeine Bearbeitungszeit des Prozesses und eventuell noch ein Ausgangspuffer.

3.3.3 Vision des simBPMN Tools

Default Prozesslogik

Um nicht für jeden Prozess eine neue Prozesslogik erstellen zu müssen, soll es in einem Editor, welcher simBPMN darstellen kann, die Möglichkeit geben, eine Defaultlogik abbilden zu können. Dieser wird automatisch an jeden Prozess angehängt und kann entsprechend vom Benutzer überschrieben werden.

Anpassbare Elemente

Um den Modellierungsprozess weiter zu vereinfachen soll es dem Benutzer möglich sein, individuelle Eigenschaften den Elementen anzufügen/ anzuhängen. Sollte dies nicht ausreichend sein, soll die Möglichkeit bestehen Dokumente zu den Elementen zu hinterlegt. In diesen können die Elemente detaillierter und in eigener Form beschrieben werden.

User Interface

Einfachheit besitzt eine grosse Priorität, deshalb sollen auf den ersten Blick, auch nur die wichtigsten Informationen in der Benutzeroberfläche dargestellt werden. Der Benutzer soll hier möglichst ohne Aufwand ein Modell abbilden können und dazu die bestmögliche Unterstützung vom Tool erhalten. Erfahrene Benutzer sollen allerdings die Möglichkeit besitzen, mithilfe eines Buttons sich alle Elemente anzeigen zu lassen.

Kapitel 4

Anforderungsspezifikationen

Die Anforderungen sind massgeblich durch die Bestimmungen der OST geprägt. Die Hauptanforderung ist es eine funktionierende Applikation zu erstellen, mit welcher ein Prozess in BPMN 2.0 modelliert werden kann. Darauf aufbauend bietet die Applikation dann die Möglichkeit simulationsbezogene Logiken ebenfalls mittels der SimBPMN abzubilden. Prozesse der realen Welt sollen so, auf eine einfache Art, digital modeliert werden können.
Das Tool soll so aufgebaut werden, dass eine unkomplizierte Erweiterung zu einem späteren Zeitpunkt jederzeit möglich ist.

4.1 Produktfunktionen

Das Tools soll die Möglichkeit besitzen, die in der simBPMN Notation definierten Elemente darzustellen. Der Anwendungsbenutzer soll dadurch die Möglichkeit erhalten, einen Prozess grafisch zu modellieren.
Die modellierten Elemente müssen die Möglichkeit besitzen, diese mithilfe von Eigenschaften detaillierter zu Beschreiben. Diese kann per Text oder auch per Dateianhang erfolgen.
Der Prozess soll in dem bereits verwendeten, auf XML basierendem, BPMN Dateiformat abgespeichert werden können. Die Datei wird um die neuen Elemente erweitert, ändert aber ihre Struktur nicht. Alle zusammengehörende Dateien sollen in einer komprimierten Form vom Tool exportiert und importiert werden können. Das Tool soll die Möglichkeit anbieten den Speicherort der Dateien festzulegen und zu ändern.

4.2 Lieferumfang

Der Lieferumfang für diese Arbeit ist ein funktionierende Applikation, welche die beschriebenen Produktfunktionen aufweist. Das erstelle Tool, sowie der Source-Code wird im Netzwerk der OST öffentlich zugänglich gemacht, um Weiterentwicklungen anzustreben.
Zusätzlich zur Applikation enthält der Lieferumfang ein Kompendium, welches beschreibt an welchen Schrauben gedreht werden muss, um gewünschte Veränderungen am Tool durchzuführen. Die Ergebnisse unserer Bachelorarbeit werden in dieser Arbeit dokumentiert und im Auftrag der Schule ebenfalls auf einem A3 Plakat festgehalten, welches nach der Arbeit ausgestellt werden soll.
Diese Punkte sowie diverse Formale Punkte (Eigenständigkeitserklärung, Glossar, Referenzen, usw..) werden in digitaler Form über das Arbeitsverwaltungstool (AVT) der OST abgegeben.

4.3 Annahmen, Einschränkungen und Abhängigkeiten

4.3.1 Annahmen

Es bestehen keine speziellen Annahmen für die Anforderungen.

4.3.2 Einschränkungen

Die Vision des simBPMN Tools welches komplett und sauber in einer Simulationsumgebung integriert ist, können wir im Umfang der Bachelorarbeit nicht erfüllen. Mit dieser Arbeit wird lediglich ein Grundstein mit minimalen Anforderungen gesetzt, welche es ermöglichen Prozesse in einem Simulationsumfeld zu beschreiben.

4.3.3 Abhängigkeiten

Die Entwicklung der Applikation ist von keiner weiteren Arbeit abhängig.

4.4 Use Cases

4.4.1 Use Case Diagramm

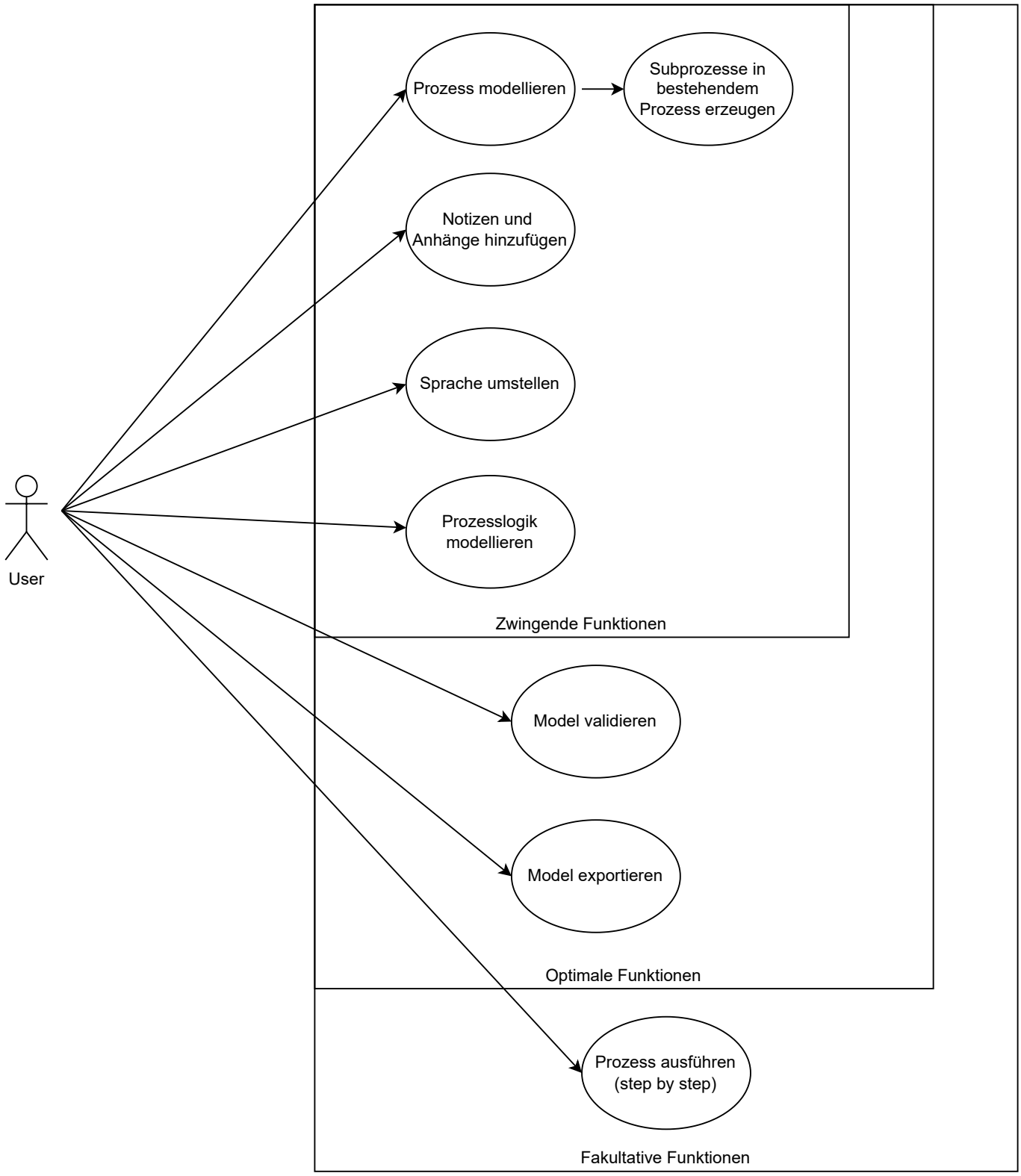


Bild 4.1: Use Case Diagramm

4.4.2 Beschreibungen (Fully Dressed)

Die nachfolgenden Use Cases im Fully Dressed Format entsprechen dem Funktionsumfang, welche ein Tool oder Framework mitbringen sollte, um weiter evaluiert werden zu können.

Prozess Modellierung

Use Case Abschnitt	**Beschreibung**
ID	1.1
Umfang	Minimal Viable Product
Level	Benutzerinteraktion
Primärer Akteur	Benutzer
Stakeholder und Interessierte	Benutzer: Will einen Prozess im Tool modellieren.
Vorbedingungen	—
Erfolgsgarantie	• Der Benutzer kann seinen Prozess erfolgreich der vom Tool gegebenen Notation abbilden.
Hauptszenario (erfolgreich)	1. Benutzer öffnet einen bestehenden oder neuen Prozess. 2. Die Software zeigt dem Benutzer die möglichen Elemente an. 3. Der Benutzer modelliert seinen Prozess mit Drag & Drop Aktionen.
Erweiterungen	—
Spezielle Bedingungen	—
Technologie- und Datenvariationen	—
Häufigkeit des Auftretens	Da dies ein notwendiger Schritt zum Erreichen einer der Kernfunktionalitäten der Software ist, tritt dieser relativ häufig auf.
Verschiedenes	—

Tabelle 4.1: Use Case: Prozess Modellieren

Subprozesse

Use Case Abschnitt	**Beschreibung**
ID	1.2
Umfang	Minimal Viable Product
Level	Benutzerinteraktion
Primärer Akteur	Benutzer
Stakeholder und Interessierte	Benutzer: Will einen Subprozess abbilden
Vorbedingungen	• Der Benutzer hat bereits einen Prozess modelliert.
Erfolgsgarantie	• Nach erfolgreichem Öffnen eines bestehenden Prozesses, kann darin ein weiterer Prozess definiert werden.
Hauptszenario (erfolgreich)	1. Der Benutzer öffnet einen bestehenden Prozesses. 2. Der Benutzer wählt das Subprozess Element in der Toolbox aus und zieht dieses in den bestehenden Prozess. 3. Die Software bettet das Element in der Zeichnung ein und ermöglicht die Modellierung des Subprozesses.
Erweiterungen	—
Spezielle Bedingungen	• Dieser UseCase ist nur in einem bestehenden Prozess möglich. Ein Subprozess kann nicht für sich alleine stehen.
Technologie- und Datenvariationen	—
Häufigkeit des Auftretens	Da dies ein notwendiger Schritt zum Erreichen einer der Kernfunktionalitäten der Software ist, tritt diese relativ häufig auf.
Verschiedenes	—

Tabelle 4.2: Use Case: Nested Prozesse

Notizen und Anhänge

Use Case Abschnitt	Beschreibung
ID	1.3
Umfang	Minimal Viable Product
Level	Benutzerinteraktion
Primärer Akteur	Benutzer
Stakeholder und Interessierte	Der Benutzer: • will einen Notiz hinterlegen. • ein File hinterlegen.
Vorbedingungen	• Der Benutzer hat bereits einen Prozess welcher dieser beschreiben möchte.
Erfolgsgarantie	• Notizen und Files können direkt aus der Zeichnung heraus aufgerufen werden.
Hauptszenario (erfolgreich)	1. Der Benutzer wählt ein Kommentar- oder Fileobjekt aus der Toolbox aus. 2. Das entsprechende Objekt kann zum beschreibenden Objekt angehängt werden. 3. Files können von der Zeichnung aus geöffnet werden
Erweiterungen	—
Spezielle Bedingungen	—
Technologie- und Datenvariationen	—
Häufigkeit des Auftretens	Da dies eine der Kernfunktionalitäten der Software ist, tritt diese häufig auf.
Verschiedenes	—

Tabelle 4.3: Use Case: Remarks und Files hinterlegen

Sprache umstellen

Use Case Abschnitt	**Beschreibung**
ID	1.3
Umfang	Minimal Viable Product
Level	Benutzerinteraktion
Primärer Akteur	Benutzer
Stakeholder und Interessierte	Der Benutzer kann die Sprache der Benutzerinterfaces anpassen.
Vorbedingungen	• Keine
Erfolgsgarantie	
Hauptszenario (erfolgreich)	Beispiel: Umstellung von Englisch nach Deutsch 1. Der Benutzer ändert im Menü die Ausgabesprache. 2. Das Interface wird nun in Deutsch angezeigt.
Erweiterungen	—
Spezielle Bedingungen	—
Technologie- und Datenvariationen	—
Häufigkeit des Auftretens	Dies ist eine Präferenz des Benutzers und wird in der Regel einmalig eingerichtet.
Verschiedenes	—

Tabelle 4.4: Use Case: Sprache umstellen

Prozesslogik modellieren

Use Case Abschnitt	Beschreibung
ID	1.3
Umfang	Minimal Viable Product
Level	Benutzerinteraktion
Primärer Akteur	Benutzer
Stakeholder und Interessierte	Benutzer: Will eine Prozesslogik modellieren.
Vorbedingungen	• Es wurde bereits ein Prozess modelliert.
Erfolgsgarantie	
Hauptszenario (erfolgreich)	1. Benutzer öffnet einen bestehenden oder neuen Prozess. 2. Die Software zeigt dem Benutzer alle verwendbaren Elemente an um eine Prozesslogik in einem Prozess zu modellieren. 3. Der Benutzer modelliert seine Prozesslogik mit Drag & Drop Aktionen.
Erweiterungen	—
Spezielle Bedingungen	—
Technologie- und Datenvariationen	—
Häufigkeit des Auftretens	Da dies ein notwendiger Schritt zum Erreichen einer der Kernfunktionalitäten der Software ist, tritt dieser relativ häufig auf.
Verschiedenes	—

Tabelle 4.5: Use Case: Prozesslogik modellieren

4.4.3 Beschreibungen (Brief)

Die nachfolgenden Use Cases im Brief Format sind nicht zwingend erforderlich für die Basisfunktionalität des bestehenden Tools oder Frameworks und besitzen eine entsprechende tiefere Priorisierung. Diese Use Cases sind eher unspezifisch und werden fortlaufend aktualisiert oder gelöscht sowie um neue Use Cases ergänzt.

Validierung eines Prozesses

Dem Benutzer soll es möglich sein, frühestmöglich Fehler in der Modellierung zu erkennen. Unter anderem sind folgende Gründe als Fehler anzusehen:

- Fehlende Start- Endpunkte
- Assoziationen von nicht kompatiblen Objekten
- Loops in Nested Processes

Step by Step Ausführung eines Prozesses

Der Benutzer kann einen modellierten Prozess Step by Step ausführen, damit ist gemeint dass im GUI ersichtlich ist, wo der Prozess steht und was in diesem Zeitpunkt passiert ist. Dies dient dazu die semantische Korrektheit zu garantieren und sollte dabei klar von einer Simulation unterscheidbar sein.

Anbindung (Export) für ein Simulationssystem

Um den abgebildeten Prozess in einem Simulationsumfeld prüfen zu können, soll es möglich sein einen Export des Modells in ein Simulationssystem zu importieren. Alternativ kann auch eine direkte Schnittstelle gesucht werden.

4.5 Nicht-funktionale Anforderungen

Verbindlichkeit

Die Verbindlichkeit der Anforderungen ist wie folgt zu interpretieren:

...muss..., ...ist zu..., ...beträgt...	zwingend, obligatorisch, hart gefordert
...sollte...	empfohlen
...kann..., ...darf...	fakultativ, erlaubt
...darf nicht...	verboten, nicht erlaubt
...muss nicht...	fakultativ, nicht zwingend

4.5.1 Funktionalität

Interoperabilität

Das Tool oder Framework welches zur Entwicklung der Visualisierungssoftware eingesetzt werden, muss auf Windows 10 und macOS ausgeführt werden können. Es muss somit beide Systeme unterstützen oder portierbar sein.

4.5.2 Zuverlässigkeit

Wiederherstellbarkeit

Die erstellten Diagramme müssen kontinuierlich vom System gespeichert werden. Das Tool oder Framework sollte somit auf einem Datenfile arbeiten. Wenn dies nicht der Fall ist, muss die Möglichkeit bestehen das Tool oder Framework um eine solche Schnittstelle zu erweitern.

4.5.3 Effizienz

Ressourcenverbrauch

Die Anwendung muss mit einem 1.5GHz Prozessor und 4 GB Arbeitsspeicher einwandfrei funktionieren.

Performance

Ein Workflow welcher bis zu 100 Elemente enthält sollte innerhalb von 3 Sekunden geladen und angezeigt werden können.

Ein Element kann einem Workflow ohne merkbare Verzögerung hinzugefügt werden.

4.5.4 Benutzbarkeit

Verständlichkeit

Das Tool oder Framework sollte die Möglichkeit bieten, die Sprache auf Deutsch oder Englisch anzuzeigen.

Bedienbarkeit

Die Benutzeroberfläche sollte auf einem 13 Zoll Monitor ohne Probleme angezeigt werden können. Die Visualisierung vom Diagramm sollte man entsprechend vergrössern oder verkleinern können.

4.5.5 Sicherheit

Für die Anwendung bestehen keine expliziten Sicherheitsanforderungen.

4.5.6 Wartbarkeit

Analysierbarkeit

Das Tool oder Framework muss eindeutig versioniert sein. Die Versionionierung muss nachvollziehbar sein und die einzelnen Versionen sollten einen Changelog beinhalten. Der Quelltext jeder Version muss in einem Versionierungssystem abgelegt sein.

Prüfbarkeit

Das Tool oder Framework sollte bereits Tests enthalten, um somit sicherzustellen, dass die vorhandene Funktionalität keine Bugs aufweist.

4.5.7 Übertragbarkeit

Installierbarkeit

Die Anwendung muss nicht installiert werden. Sie sollte direkt auf dem Rechner ausgeführt werden können.

Die Anwendung muss ohne Administratorrechte ausgeführt werden können.

Die Anwendung darf keine Einträge in die Windows Registry schreiben.

Koexistenz

Auf einem Rechner darf nur eine Instanz der Anwendung ausgeführt werden.

Kapitel 5

Analyse

5.1 Dateispeicherung

5.1.1 BPMN IO

Die Datensicherung über das BPMN IO Framework kann mithilfe des Properties Panel umgesetzt werden. Die Schnittstelle des Properties Panel erlaubt es eigene Properties zu definieren, welche anschliessend in die XML Datei des Diagramms eingebettet werden. Dabei erhaltet jedes Element in der XML Datei ein Attribut mit den entsprechenden Werten.

5.1.2 Konzeptideen

Für die Funktionalität der Dateispeicherung wurden drei verschiedenen Ansätze entwickelt. Diese wurden analysiert und anschliessend zusammen mit dem Betreuer besprochen. Schlussendlich wurde aus den drei Ansätzen ein passender für die Umsetzung ausgewählt.

Verlinkung

Bei der Verlinkung übernimmt die Applikation keine Verantwortung für die entsprechenden Dateien. Die Applikation bezieht sich auf den aktuellen absoluten Pfad der entsprechenden Dateien. Der Entwicklungsaufwand in diesem Ansatz ist minimal, allerdings besteht weder von der Applikation, noch von dem Benutzer eine Kontrolle, ob die Dateien in der Applikation verwendet werden.

Vorteile	Nachteile
• Geringer Entwicklungsaufwand • Keine Kopien der Dateien werden erstellt	• Fehleranfälligkeit durch Benutzeränderungen am Dateisystem (Verschieben der Dateien) • Keine Übersicht welche Dateien von der Applikation verwendet werden

Tabelle 5.1: Vor- und Nachteile von Verlinkung

Dateiordner am Speicherort

Um Dateien unabhängig vom Benutzer verwenden zu können und die Fehleranfälligkeit zu verringern, besteht die Möglichkeit, dass die Applikation am Speicherort der Modelldatei einen Ordner erstellt und alle verlinkten Dateien in diesen hinein kopiert werden. Dieser Ansatz bietet dem Benutzer die Sicherheit, dass er Dateien auf seinem System weiterhin bewegen und verändern kann, ohne dass die Modelldatei diese Dateien nicht mehr auffinden kann. Andererseits muss für die Datei eine Kopie angefertigt werden, wodurch das Speichersystem des Benutzers belastet wird und allenfalls unbewusste Kopien entstehen.

Vorteile	Nachteile

• Übersicht der verwendeten Dateien im Filesystem • Verringerte Fehleranfälligkeit	• Erstellung von Kopien • Die Modelldatei muss bevor Dateien angehängt wurden einmal gespeichert werden.

Tabelle 5.2: Vor- und Nachteile des Dateiordners

Workspace

Eine Erweiterung eines einfachen Ordners am Speicherort der Modelldatei, ist das Konzept eines Workspaces. In einem Workspace werden alle Dateien, welche in der Applikation verwendet werden gespeichert. Der Workspace besitzt in der Standardeinstellung den Namen `simBPMN_workspace` und wird im Benutzerordner des entsprechenden Benutzers abgelegt. Auf den entsprechenden Systemen ist dieser wie folgt definiert:

Windows: `C:\\Users\\<Username>`

OSX: `/Users/<Username>`

Linux: `/home/<Username>`

Auf der obersten Ebene des Workspaces sind die Modelldateien abgelegt. Für jede Modelldatei wird von der Applikation ein Unterordner, mit dem Namen der Modelldatei generiert. Alle Anhänge werden in den entsprechenden Unterordner gespeichert.
Durch diesen Ansatz sind alle Dateien welche die Applikation verwendet an einem Ort gespeichert. Es besteht die Möglichkeit, mit diesem Ansatz alle Dateien in einem Panel in der Applikation anzuzeigen. Der Benutzer kann somit einfach und schnell alle seine Modelldateien öffnen. Durch die entstehende Struktur wird allerdings mehr Entwicklungsaufwand benötigt.

Vorteile	Nachteile
• Benutzer hat Übersicht über Diagramme. • Alle Anhänge sind an einem Ort zu finden	• Höherer Entwicklungsaufwand

Tabelle 5.4: Vor- und Nachteile des Workspaces

Fazit

Die folgende Tabelle bietet eine Übersicht, wie die verschiedenen Konzepte in Zusammenhang stehen. In der Besprechung C.5 vom 29.März mit Prof. Dr. Andreas Rinkel wurde beschlossen, die Variante mit dem Workspace zu verwenden und diese Umzusetzen.

Kategorie	Verlinkung	Dateiordner	Workspace
Aufwand / Komplexität	Sehr niedrig	Niedrig	Erhöht
Fehleranfälligkeit	Hoch	Mittel	Niedrig
User Experience	Niedrig	Mittel	Erhöht

Tabelle 5.5: Übersicht der Varianten

5.1.3 Import & Export

Um Dateien in einem anderen Visualizer importieren zu können, muss vorerst ein Export erstellt werden. Die Exportfunktion sammelt dafür alle Dateien im Workspace, welche mit der Datei im Zusammenhang stehen und erstellt mit der Modelldatei zusammen eine komprimierte ZIP Datei. Beim Import legt die Applikation

die komprimierte Datei im entsprechenden Workspace ab.

5.1.4 User Interface

Im User Interface besteht unter dem Menüpunkt "Einstellungen" die Möglichkeit den Pfad und Ordnernamen für den Workspace zu ändern. Dateien selber können im Properties Panel des Elements hinzugefügt werden. Derzeit besteht noch keine visuelle Anzeige auf dem Modell um erkennen zu können, ob eine Datei hinzugefügt wurde.

Kapitel 6

Ergebnisse

Das Resultat dieser Arbeit besteht aus dem Spezifikationstool, sowie das Kompendium in welchem erklärt wird wie man mit den verschiedenen eingesetzten Frameworks umgeht.

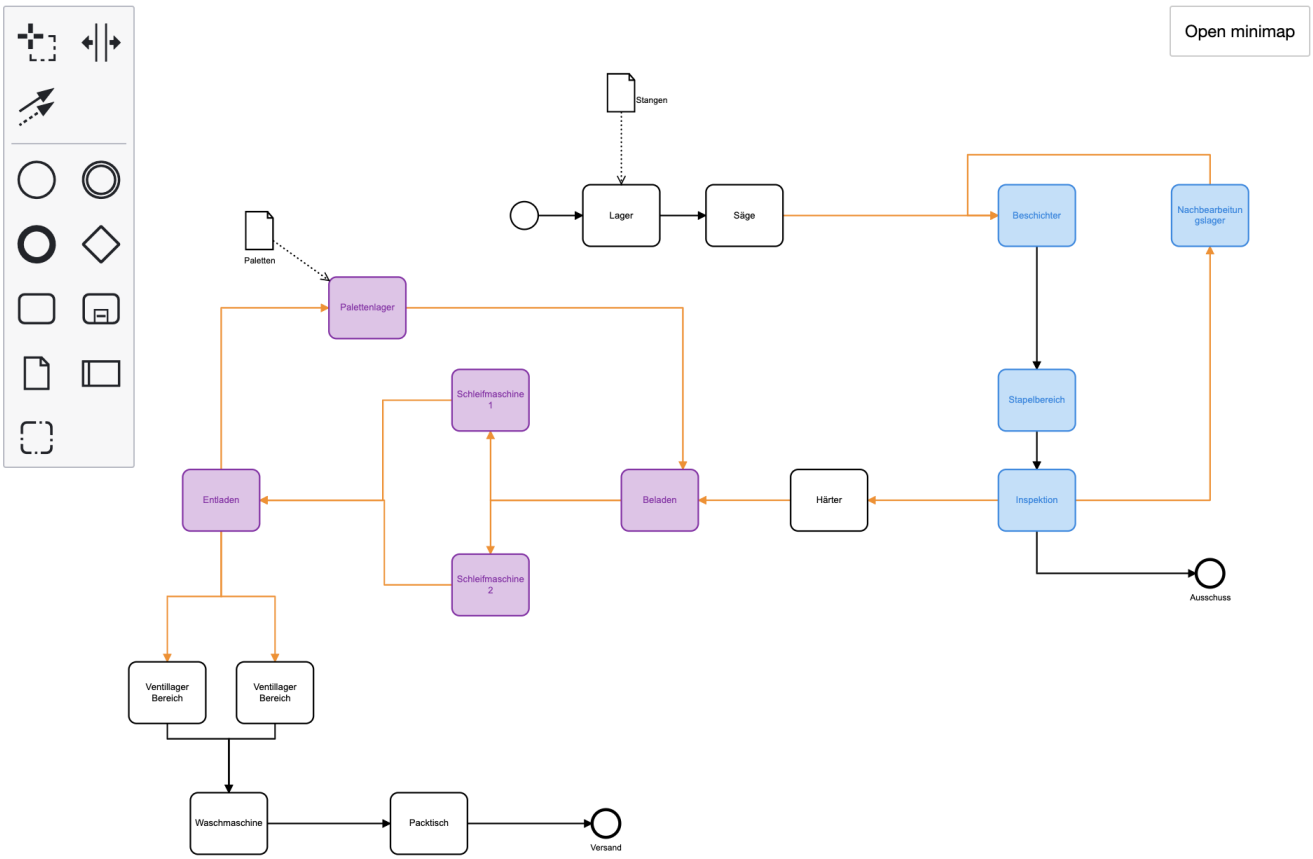
6.1 Nicht funktionelle Anforderungen

Die nicht funktionellen Anforderungen konnten alle erfüllt werden. Das Tool ist performant und erfüllt dank dem Electron Framework verschiedenste Anforderungen. Wir sind während der Entwicklung auf keine Anforderung gestossen, welches ein Problem dargestellt hat.

6.2 Modeler

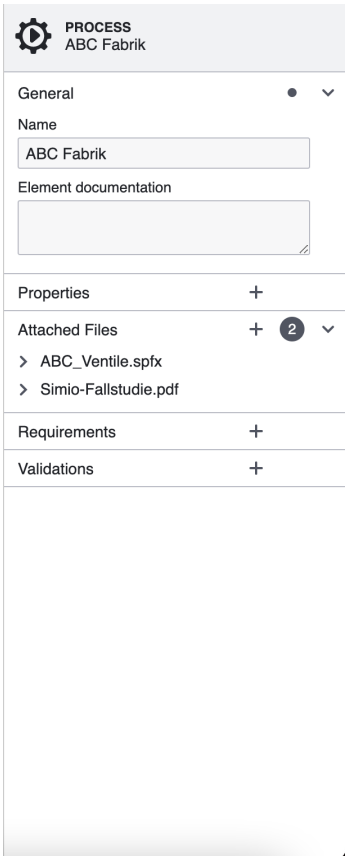
Der fertige Modeler besteht aus verschiedenen funktionellen Anforderungen welche im folgenden aufgefasst werden.

6.2.1 Canvas



Der Canvas ist das Herz des Modelers. In diesem können wir Prozesse spezifizieren und besitzen die Fähigkeit die definierten simBPMN Elemente zu erstellen und mit einander zu verbinden. Der Canvas basiert auf dem BPMN IO Framework und wurde von uns um verschiedene Funktionalitäten und Elemente erweitert. Der Canvas bietet zudem die Möglichkeit in Subprozesse einzutauchen und diese weiter zu spezifizieren. Zusätzlich lässt sich der Canvas teilen um die Logik der verschiedenen Teilprozesse spezifizieren zu können.

6.2.2 Properties Panel



Das Properties Panel arbeitet stark mit dem Canvas zusammen. Das Properties Panel zeigt immer für das derzeit ausgewählte Element die Eigenschaften an. Das Properties Panel kann somit detaillierter die verschiedenen Elemente beschreiben. Sollte die Möglichkeiten nicht ausreichen, hat der Benutzer auch die Möglichkeit die Elemente in einem anderen Format zu beschreiben und die Beschreibung als Anhang im Properties Panel hinzuzufügen.

6.2.3 Workspace

Der Workspace beinhaltet alle Dateien welche der Modeler derzeit verwenden kann und wo er diese auch abspeichert. Alle Änderungen welche im Canvas oder im Properties Panel vorgenommen werden werden im zugehörigen BPMN File abgespeichert, welches sich im Workspace befindet. Alle Dateien des Workspaces können exportiert und importiert werden um somit den Austausch zwischen verschiedenen Rechnern zu gewährleisten.

6.2.4 Ausblick

Die Applikation wird nach dem Abschluss der Arbeit weiter gewartet und erweitert. Das Spezifikationstool bietet sehr viele Stellen, an welchem noch weitere Funktionalitäten eingeführt werden können. Es bestehen bereits Ideen die Spezifikation zu validieren und auf verschiedene Fehler zu überprüfen. Ein Durchlaufen vom Prozess und wie sich dieser Verhält, ist auch bereits im Backlog aufgenommen. Das Tool bietet viel Potenzial zur Weiterentwicklung.

Kapitel 7

Schlussbericht

7.1 Zielerreichung

Das Ziel der Bachelorarbeit war es die vorangegangene Studienarbeit mit den evaluierten Technologien umzusetzen. Alle dafür benötigten Komponenten sollten gemäss Aufgabenstellung Open-Source sein, damit die zu entstehende Applikation nach Beendigung der Bachelorarbeit frei verfügbar gemacht werden kann und Interessierte darauf weiterentwickeln können. Der Fokus der Arbeit besteht darin einen Grundstein zu legen auf welchem weiterentwickelt werden kann. Stabilität und Benutzerfreundlichkeit werden einem umfassenden Feature-Katalog vorgezogen. Die erforderte Basisfunktionalität der Applikation sieht vor, dass ein Benutzer ein Prozessmodell sauber abbilden und deren Logik beschreibend festhalten kann.
Aus unserer Sicht ist der Auftrag der Bachelorarbeit erfüllt. Wir konnten uns an unseren in der Studienarbeit zuvor erstellen Plan halten und mussten nicht davon abweichen. Der Grundstein ist gesetzt und dokumentiert. So sollten dritte Personen ebenfalls die Möglichkeit besitzen unseren Code zu verstehen.
Persönliche Erkenntnisse, die während der Arbeit gewonnen wurden, sind im darauf folgenden Kapitel zu finden.

7.2 Review der eingesetzten Tools

GitLab ist ein äusserst umfangreiches und gut geeignetes Tool zur Versionsverwaltung. Da darin auch Issues erstellt werden können, kann GitLab zusätzlich als Issue-Tracker verwendet werden. Der Umgang mit Git wurde bereits in vorgegangen Modulen erlernt und konnte hier ohne weiteres angewendet werden. Wir haben uns entschieden als Zeiterfassungstool YouTrack zu verwenden. YouTrack gehört zur JetBrains Familie und kann via Web benutzt werden. Wie auch für andere JetBrains Produkte üblich, ist dieses einfach in der Handhabung und Bedienung. Als Issue-Tracker wurde ebenfalls YouTrack verwendet da man so direkt Zeit auf einen Issue buchen kann.

7.3 Persönliche Erfahrungen

In den nachfolgenden Abschnitten erläutert jedes Mitglied seine Erfahrungen während der Bachelorarbeit eigenständig und ohne Korrelation zueinander. Diese Abschnitte sind in der Ich-Perspektive geschrieben, um den persönlichen Aspekt zu verstärken.

7.3.1 Michel Mirsayyah

7.3.2 Vor dem Projektstart

Der Einstieg in die Bachelorarbeit war für mich einmal mehr komplizierter als nötig. Ich hatte es geplant einen selber eingebrachten Themenvorschlag als Bachelorarbeit durchzuführen. Dies ist von Seiten der Schule aus auch möglich, jedoch extrem verkompliziert. Nachdem also mein Industriepartner und ich durch jeden Reifen der Schule gehüpft sind, um den Anforderungen gerecht zu werden, hiess es dass Thema sei eingereicht und die Schule meldet sich wieder bis spätestens zur vorgegebenen Deadline. Nach Ablauf der Deadline war noch keine Rückmeldung der Schule weder bei mir noch beim Industriepartner eingegangen. Einmal mehr wurde ich von der Schule ignoriert und sitzen gelassen. Als ich dann aktiv wurde und nach dem Stand der Dinge fragte, hiess es bloss: "Wenn sich bis jetzt niemand gemeldet hat, dann wirds wohl nichts mehr...". Hut ab, 1A Kommunikation von Seiten der Schule. Die Probleme einfach solange ignorieren, bis sie keine mehr sind. Zu meinem Glück war die Stelle der Folgearbeit der SA noch frei. So konnte ich wieder zusammen mit Sven arbeiten.

7.3.3 Allgemeines zum Projekt

Sven und ich haben als Vorarbeit eine Evaluation des zu erstellenden Tools gemacht, die wir in diesem Projekt umgesetzt haben. Dadurch hatten wir schon eine Ahnung um was es sich dabei handelt und konnten so relativ zeitnah mit dem Implementieren beginnen.

7.3.4 Das Projekt

Wie vorhin schon genannt kannten wir den Rahmen der Aufgabenstellung, da wir diesen Teilweise auch selber definiert haben. Die eingesetzten Technologien wurden in der Studienarbeit gut evaluiert und passten auf unseren Anwendungsfall. Es war nicht nötig dort Änderungen vorzunehmen, dementsprechend konnte die Implementation wie geplant ausgeführt werden. Das Projekt verlief aber nicht ohne Probleme, der Setup oder Barebone des Projekt hat relativ viel Zeit und Energie beansprucht. Hier mussten teilweise aufgrund mangelnder Dokumentation einfach Einstellungen ausprobiert werden, bis diese dann gepasst haben. Während der Implementationsphase gab es auch kleine Hürden, die uns auf die Probe gestellt haben. Für mich waren die grösseren Probleme auf Seiten von BPMN IO. Das Framework ist zwar super, jedoch existiert noch keine benutzbare Dokumentation dazu. Es gibt dafür jeweils Beispielprojekte in welchen gezeigt wird, wie etwas umgesetzt werden kann. Das Electron Framework ist fast das gegenteil zu BPMN IO. Hier gibt es eine gute Dokumentation, jedoch hatte ich teilweise Fehler wenn ich gemäss den offiziellen Docs etwas implementiert habe. Schlussendlich konnten aber alle angetroffenen Probleme gelöst werden ohne dass die Codebasis dafür verletzt werden mussten. Das Tool ist meiner Meinung nach so geworden wie geplant und läuft stabil. Wie versprochen kann alles am Tool ergänzt oder verändert werden, die nötige Dokumentation liegt als Kompendium vor. Zudem wurde das ganze Projekt aus OpenSource Komponenten erstellt und kann deshalb öffentlich Verfügbar gemacht werden. Das Tool sollte als ersten Wurf angesehen werden und als Grundstein für weitere Features dienlich sein.

7.3.5 Persönliches

Auf Zwischenmenschlicher Ebene gab es keine Probleme im Projekt. Wir konnten alle sowohl im Team als auch Autonom arbeiten und hielten uns gut an die Abgemachten Ziele. Ich hatte Spass an der Arbeit möchte aber nochmals betonen, dass ein solches Verhalten bezüglich der Kommunikation der Schule nicht in Ordnung ist.

7.3.6 Sven Höpfner

Durch die vorangehende Studienarbeit wusste ich bereits was ungefähr auf mich zu kommt und war selbstbewusst, dass wir die bevorstehenden Hürden meistern können. Selber hatte ich bisher noch nicht viel Erfahrung mit Javascript sammeln können und hatte daher vor diesem Punkt am meisten Respekt. Über die Arbeit hinweg konnte ich mich jedoch in die Sprache sehr gut einarbeiten und bin zuversichtlich, dass ich in einem vernünftigen Stil meine Arbeit erledigen konnte. Ich bin zufrieden mit den Entscheidungen die wir für das Projekt getroffen haben. Im Bereich der Technologien hatten wie mit Electron einige Startschwierigkeiten, sobald die aber überwunden wurden, gab es keine weiteren Auffälligkeiten. In das BPMN Framework konnte ich mich ebenfalls gut einarbeiten und dank den vielen Foreneinträgen, in welchen Fragen oder auch Tipps zu dem Framework gepostet wurden, sehr gut zurecht finden. Die Zusammenarbeit mit meinem Partner und dem Betreuer hat mir sehr gut gefallen und ich habe diese positiv in Erinnerung. Schlussendlich denke ich, dass ich von der Arbeit einiges in mein berufliches Leben mitnehmen kann.

Kapitel 8

Zeitauswertung

Unten aufgeführt sind Diagramme, welche den verbrauchten Zeitbedarf darstellen. Diese sollen einen aufgeschlüsselten Einblick über den erbrachten Aufwand über die gesamte Projektzeit der Bachelorarbeit visuell darstellen. Wie an den unten stehenden Diagrammen zu erkennen, gab es keine aussergewöhnliche Abweichungen im erbrachten Aufwand. Das Projekt lief mehrheitlich wie geplant und es gab keine Hindernisse, die das Projekt in irgend einer Form beeinträchtigt haben.

8.1 Zeitauswertung pro Person

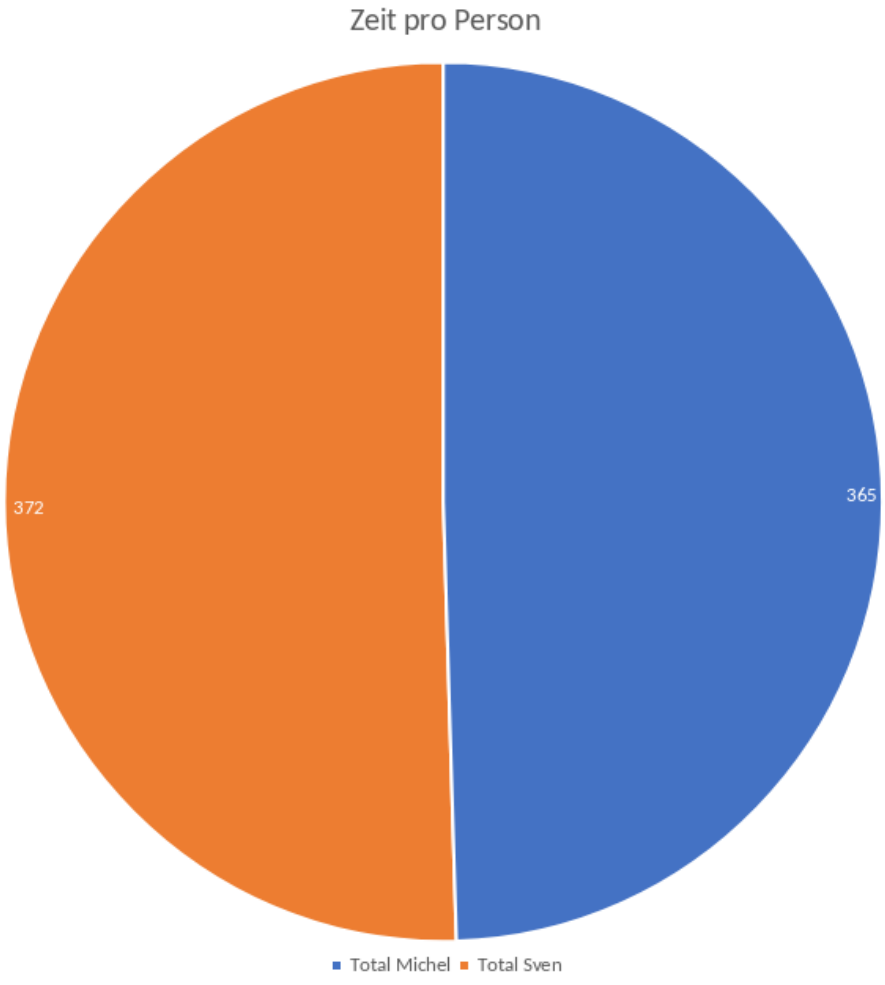


Bild 8.1: Zeitauswertung pro Person

8.2 Zeitauswertung pro Person und Woche

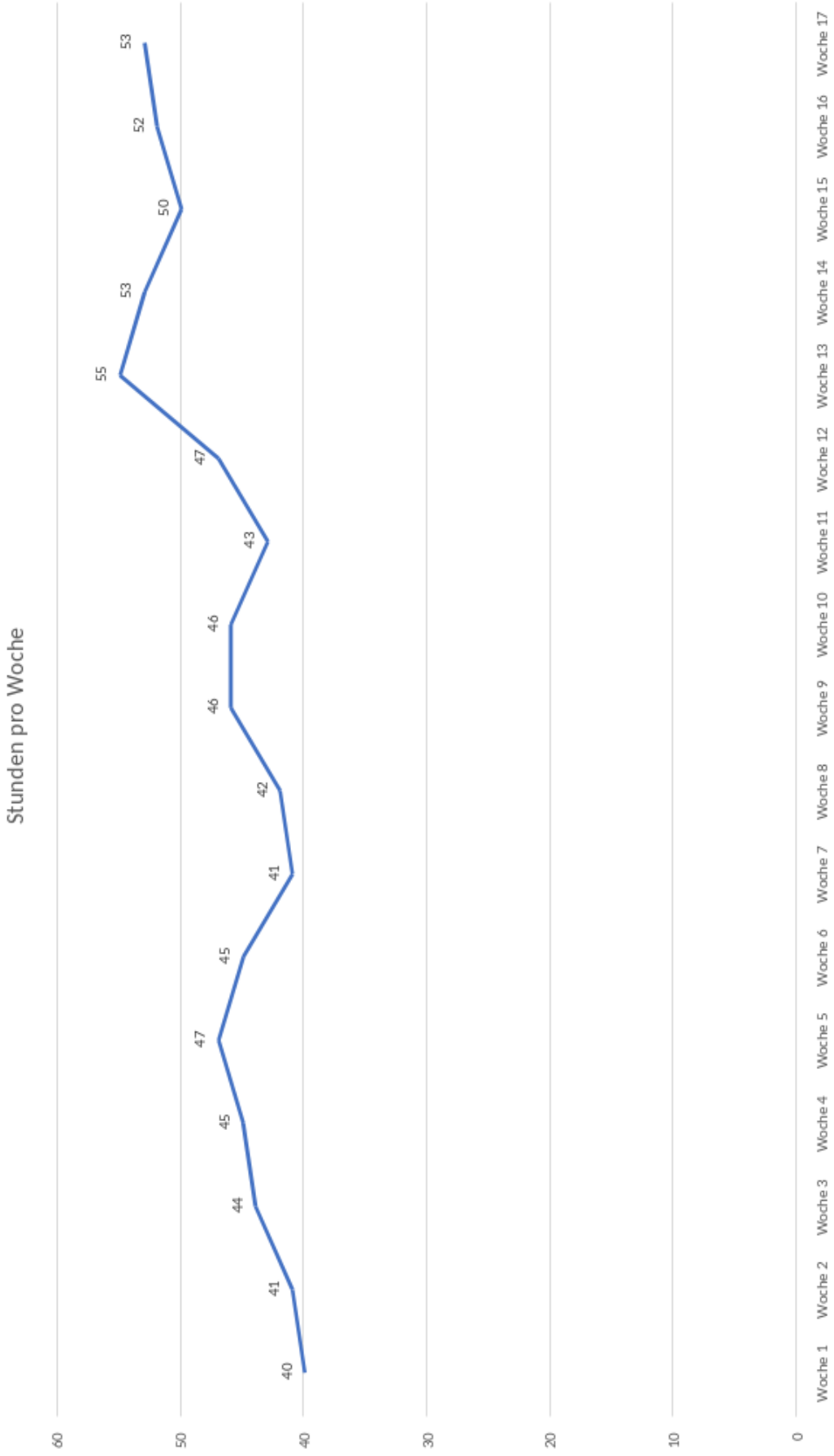


Bild 8.2: Zeitauswertung pro Person und Woche

Kapitel 9

Referenzen

[1] *Bpmn 2.0 syntax & semantik*, http://wi-wiki.de/doku.php?id=prozessmodellierung:modellierungshaus:bpmn_2.0_syntax_und_semantik.

[2] Awesome Open Source, *Liste von open source bpmn projekten*, https://awesomeopensource.com/projects/bpmn.

[3] Interprocess Kommunikation, *Beschreibung von medium.com*, https://medium.com/teneocto/electronjs-how-to-communicate-between-main-process-renderer-process-and-injected-webview-1b4fbd76e0b7.

[4] CodePen, *Codepen.io - website mit html snippets*, https://codepen.io.

[5] Sikriti Dakua, *Author des code-snippets*, https://dribbble.com/devloop01.

[6] Sikriti Dakua - Code-Snippet, *Code-snippet von codepen*, https://codepen.io/dev_loop/pen/WNvBzZG.

[7] *Electron framework*, https://www.electronjs.org/.

[8] *Bpmn io*, https://bpmn.io/.

Kapitel 10

Glossar

ASIM
: Arbeitsgruppe Simulation.

BPMN IO
: Javascript Toolkit entwickelt durch Camunda Services.

BPMN 2.0
: Business Process Model and Notation.

Electron
: Electron ist ein von GitHub entwickeltes Framework um Plattformunabhängige Applikationen zu erstellen..

Electron Forge
: Electron Forge ist eine Art Erweiterung für Electron, welche erweiterte Funktionen wie beispielsweise das erstellen eines Installers mit sich bringt..

Incpetion
: Die Inception Phase beschreibt in einem Projekt die Ausarbeitung der Vision.

Jira
: Issue Tracker- sowie Zeiterfassungstool von Atlassian - https://simbpmn.atlassian.net.

OST
: Ostschweizer Fachhochschule.

Scrum
: Scrum ist ein Vorgehensmodell des Projekt- und Produktmanagements, insbesondere zur agilen Softwareentwicklung. Es wurde ursprünglich in der Softwaretechnik entwickelt, ist aber davon unabhängig.

Scrum+
: Eine Vermischung des Wasserfallmodels und Scrum. Gearbeitet wird nach Scrum, ergänzt um Meilensteine aus dem Wasserfallmodell.

simBPMN
: Erweiterte Business Process Model and Notation.

Systems Modeling Language
: Grafische Modellierungssprache mit welcher komplexe Systeme abgebildet werden können.

XML
: Die Extensible Markup Language, abgekürzt XML, ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten..

YouTrack
: Issue Tracker- sowie Zeiterfassungstool von JetBrains.

Kapitel A

Projektplan

A.1 Projektmanagement

A.1.1 Organisationsstruktur

	Verantwortung	Entwicklung, Analyse, Architektur
	Name	Mirsayyah
	Vorname	Michel
	OST-Email Adresse	michel.mirsayyah@ost.ch
	Erfahrung	Softwareentwicklung .NET und Java
	Verantwortung	Entwicklung, Dokumentation (Zeiterfassungskontrolle, Design, Protokolle)
	Name	Höpfner
	Vorname	Sven
	OST-Email Adresse	sven.hoepfner@ost.ch
	Erfahrung	Softwareentwicklung .NET und Java

Tabelle A.1: Projektorganisation und Verantwortung

Name	Funktion
Prof. Dr. Andreas Rinkel	Betreuer
Knut Schmahl	Experte
Ivan Bütler	Gegenleser

Tabelle A.2: Externe Personenschnittstellen

A.2 Managementabläufe

A.2.1 Zeitbudget

Dem Modul "Bachelorarbeit" werden 12 ECTS-Punkte angerechnet. Dies entspricht umgerechnet pro Person ein Budget von 360h. Die Bachelorarbeit wird während des Frühjahrssemester 2022 durchgeführt. Das Semester besitzt bis zur Abgabe der Arbeit 17 Wochen. Mit einer gleichmässigen Verteilung ergibt sich somit ein Pensum von ca. 22h pro Woche pro Student.

Start-Termin: 21.02.2022 **End-Termin:** 17.06.2022

A.2.2 Zeitliche Planung

Das Projekt wird mit dem erlernten Model Scrum+ durchgeführt. In diesem werden die Phasen des Rational Unified Process Models mit Scrum kombiniert. Entsprechend werden während der verschieden Phasen, Sprints von jeweils 2 Wochen durchgeführt. Diese Sprints werden auf 5 Meilensteine aufgeteilt. Die Meilensteine beschreiben wesentliche Punkte der Arbeit und dienen jeweils zur Fortschrittskontrolle während der Arbeit. Nach dem Erreichen jedes Meilensteins wird der derzeitige Stand der Arbeit für ein Review an Prof. Dr. Andreas Rinkel übergeben.

Der Zeitplan für diese Arbeit beinhaltet bewusst keine Incpetion Phase. Dies ist dadurch begründet, dass die in der Inception Phase vorgesehende Vision bereits in der Vorarbeit definiert wurde.

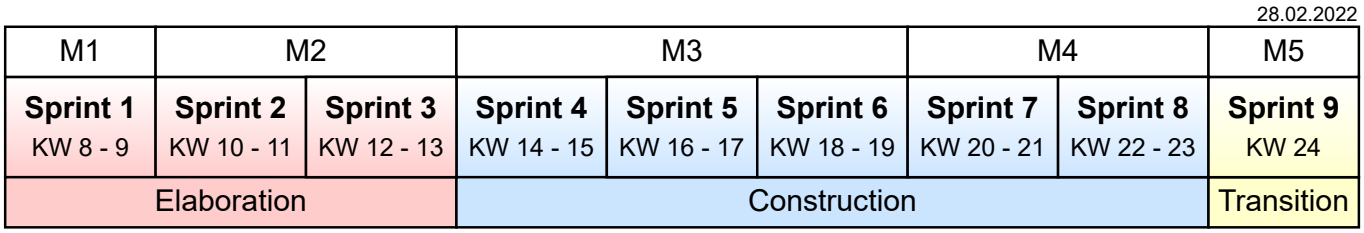


Bild A.1: simBPMN Zeitplan

Meilensteine

Nummer	Bezeichnung	Erreicht am	Beschreibung
M1	Projektstart	06.03.2022	Projektplan und Anforderungsspezifikation erstellt und mit Prof. Dr. Andreas Rinkel besprochen.
M2	End of Elaboration	03.04.2022	Beschluss, ob eine eigene Applikation aufgebaut oder der Camunda Modeler zurückgebaut wird. Ein Prototyp der entsprechenden Applikation wurde erstellt.
M3	Basisapplikation	15.05.2022	Die Basisapplikation wurde entwickelt. Alle definierten Basisanforderungen wurden umgesetzt.
M4	Prozesslogik	12.06.2022	Die Prozesslogik wurde ausgearbeitet und in der Basisapplikation eingebaut.
M5	Abschluss + Präsentation	17.06.2022	Das Projekt wurde abgeschlossen und die Präsentation vorbereitet.

Tabelle A.3: Meilensteine

Sprintplanung

Die Dauer eines Sprints ist, mit Ausnahme von Sprint 9, mit zwei Wochen definiert. Aufgrund der ungeraden Anzahl Wochen ist die Dauer von Sprint 9 auf eine Woche reduziert. Da in diesem allerdings nur Abschlussarbeiten getätigt werden, ist dies vernachlässigbar.

Sprint	Start	Ende	Tätigkeiten
1	21.02.2022	06.03.2022	• Managementabläufe definieren • Risikoanalyse durchführen • Arbeitspakete definieren • Anforderungsanalyse erstellen • Qualitätsmassnahmen definieren
2	07.03.2022	20.03.2022	• Verwendete BPMN Elemente definieren • simBPMN Elemente definieren • Electron mit BPMN IO aufsetzen • Einfaches User Interface • Statusbesprechung mit Universität München
3	21.03.2022	03.04.2022	• Electron Projekt aufsetzen • Speicherkonzept ausarbeiten • User Interface definieren
4	04.04.2022	17.04.2022	• Properties Panel einbauen • Mehrsprachigkeit einbauen • Menu Elemente anpassen
5	18.04.2022	01.05.2022	• Properties Panel erweitern • Seitenpanel einbauen • Farbauswahl einbauen
6	02.05.2022	15.05.2022	• Pallete individualisieren • Context Pad individualisieren • Seitenpanel erweitern
7	16.05.2022	29.05.2022	• Splitview einbauen • SimBPMN Elmente hinzufügen • Einstellungen abspeichern
8	30.05.2022	12.06.2022	• Workspace exportieren & importieren • Einstellungsseite hinzufügen

9	13.06.2022	17.06.2022	• Dokumentation fertigstellen • Plakat erstellen • Präsentation vorbereiten • Arbeit abgeben

Tabelle A.4: Sprints

Sonstiges

Während der gesamten Projektdauer wird die Dokumentation entsprechend erweitert und aktualisiert. Der Stand dieser wird am Ende jedes Sprints überprüft.
Bei allfälligen Problemen oder Besonderheiten wird dies in der wöchentlichen Besprechung besprochen und im Meetingprotokoll festgehalten.

A.2.3 Besprechungen

Es werden wöchentliche teaminterne Meetings abgehalten, diese entsprechen dem Weekly des Scrum Models. An diesen Meetings werden der aktuelle Stand der Arbeit, mögliche Schwierigkeiten, sowie die bevorstehenden Punkte ausgetauscht und besprochen.

Zusätzlich wird wöchentlich ein Meeting in Person mit dem Team und dem Betreuer auf dem Gelände der OST abgehalten. In diesem wird der Stand sowie das weitere Vorgehen besprochen. Die Besprechungen werden protokolliert. Auf diese Meetings folgen jeweils teaminterne Meetings, welche als Sprint Planning, Retroperspective sowie als Refinement Meeting dienen.

Ausserhalb der geplanten Meetings sind beide Gruppenmitglieder, sowie der Betreuer offen für zusätzliche Besprechungen, sollten diese von Nöten sein.

A.3 Risikomanagement

A.3.1 Risikoanalyse

Mit der Risikoanalyse wird sichergestellt, dass die Bachelorarbeit über Strategien verfügt, um mit diversen Risiken umgehen zu können. Die Analyse beinhaltet für jedes Risiko ein Strategie um dieses Vorzubeugen und mit diesem umgehen zu können. Durch diese Risikoanalyse wird einem Scheitern des Projekts entgegengewirkt.

Nr	Risiko	Beschreibung	max. Schaden	Eintrittswahrscheinlichkeit	Vorbeugung	Verhalten beim Eintreten
R1	Abhängigkeiten des Camunda Modeler	Bei der Umsetzung des Camunda Modelers werden unbemerkte Abhängigkeiten entdeckt, welche entfernt und ersetzt werden müssen.	48h	40%	In der Vorauswahl werden die Abhängigkeiten direkt angeschaut und analysiert. Die entsprechenden anstehende Arbeit wird dokumentiert und in die Arbeitsplanung mit einbezogen	Der Aufwand der anstehenden Arbeit wird geschätzt und in die Arbeitsplanung mit aufgenommen.
R2	Bugs und Fehler	Schwerwiegende funktionelle und konzeptionelle Fehler werden erst spät entdeckt und führen zu Probleme	24h	10%	Die durchgeführten Änderungen werden mithilfe eines Merge-Requests gegenseitig überprüft.	Bei auftretenden Fehlern sind diese schnellstmöglich zu beheben. Dies führt dazu, dass diese nicht lange existieren und Folgeprobleme umgangen werden.
R3	Konflikt im Team	Meinungsverschiedenheiten führen zu Streit.	16h	10%	Direkte und offene Kommunikation	Eskalation und Streitschlichtung durch den Betreuer.
R4	Krankheit	Teammitglieder fallen aufgrund von Krankheiten aus.	32h	20%	Die Besprechungen werden protokolliert.	Das abwesende Mitglied wird von dem anderen Partner auf dem neusten Stand gehalten. Falls es dem Mitglied nicht möglich ist, an dem Projekt währenddessen weiterzuarbeiten, soll eine Umverteilung der Arbeitspakete in betracht gezogen werden.

Tabelle A.5: Risikomanagement

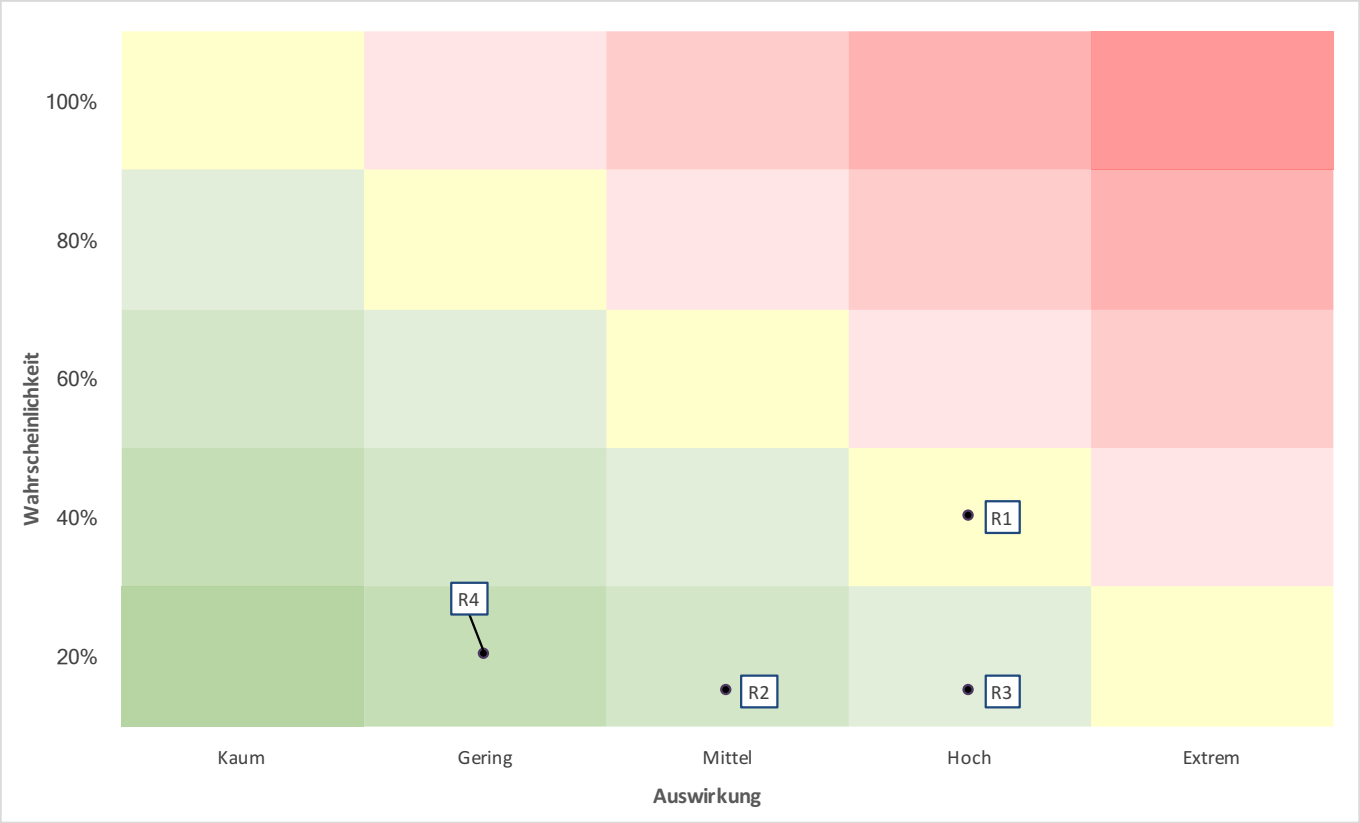


Bild A.2: Projektrisikien Übersicht

A.3.2 Umgang mit Risiken

Qualitätssichernde Überprüfungen: Nach jedem Sprint werden die Risiken neu evaluiert, sodass eine hohe Qualität des Risikomanagements während der gesamten Projektdauer garantiert werden kann.

A.3.3 Wichtige Termine

Datum	Termin
22.04.2022	Zwischenpräsentation der Bachelorarbeit
24.06.2022	Abgabe der Bachelorarbeit
28.06.2022	Präsentation der Bachelorarbeit

Tabelle A.6: Wichtige Termine

A.4 Arbeitspakete

Arbeitspakete umfassen Epics, User Stories und Tasks. Sie werden als Issues in Jira erfasst und durch Labels unterschieden. Die Issues werden stets verfeinert und aktuell gehalten. Die Issues werden auf beiden Projekten, Code: BA-Arbeit und Dokumentation: BA-Dokumentation, geführt. Der gesamte Produkt-Backlog setzt sich demnach aus allen Issues der beiden Projekte zusammen und befindet sich im Hauptprojekt: Gruppe BA.

Nr.	Arbeitspaket	Beschreibung	Zeit (h)	Priorität
1	Projektplan erstellen	Projektplan erstellen und ausarbeiten	24h	1
2	Grundgerüst bereitstellen	Grundgerüst zur Implementation bereitstellen	24h	1
3	Planung Architektur	Softwarearchitektur für die Implementation planen	48h	1
4	UI Konzept erstellen	User Interface planen	16h	1
5	Speicherungskonzept erstellen	Konzept planen	16h	1
6	Cleanup Codebasis	Vorbereitung für Implementation	16h	1
7	UI implementieren	User Interface implementieren	32h	1
8	Speicherungskonzept implementieren	Konzept implementieren	24h	1
9	Multilingualität implementieren	Funkionen für ein Mehrsprachiges UI erstellen	8h	1
10	Prozesslogik implementieren	simBPMN-Erweiterung für Prozesse	64h	1
11	Testing	Integrations & Unit Tests	56h	1
12	Administrativer Aufwand	Dokumentation, Administration, Meetings	288h	1
13	Präsentationen	Zwischen- sowie Abschlusspräsentation	64h	1
14	Indikator für Elemente	Indikator zeigt an ob Prozesse vom Default abweichen	32h	2
15	Subprozesse öffnen	Subprozesse können separat betrachtet/geöffnet werden	8h	2
16	Easy & Expanded Mode der Elemente	Die Elemente sollen je nach dem welcher Modus ausgewählt wurde Elemente ausblenden oder nicht	8h	2

Tabelle A.7: Arbeitspakete

Nr.	Begründung / Berechnung der Zeit
1	Erstellung der Anforderungsspezifikation sowie des Projektplan der Bachelorarbeit.
2	Die Codebasis aufräumen und updaten. Dies ermöglicht es sauber darauf aufzubauen.
3	Für die Planung der Architektur der Applikation und ihrer verschiedenen Funktionalitäten, sowie deren Zusammenarbeit werden 6 Tage eingeplant. Es wurden bewusst pro Person 3 Tage gewählt, damit genug Zeit besteht allfällige Fehler von Anfang an vermeiden zu können.
4	Die Planung des User-Interfaces soll im Team besprochen werden können. Mit den Budgetierten 3 Tagen kann so ein Teammitglied einen Vorschlag erarbeiten, welcher anschliessend im Plenum besprochen werden kann.
5	BPMN IO verwendet zur Persistierung das standardisierte BPMN Format. Es wurden 3 Tage für das Speicherkonzept einberechnet, da zu dem Format noch keine Erfahrung besteht. Es muss zuerst ein Plan erstellt werden, wie der Simulationsteil darin ablegt werden kann.
6	Um die CodeBasis auf einen Stand zu bringen auf welcher danach aufgebaut werden kann, werden 2 Tage berechnet.
7	Für die Implementierung der User Interfaces werden 4 Tage budgetiert. Die Zeit soll verwendet werden um das Konzept umzusetzen und Tests zu erstellen.
8	Für die Implementierung des Speicherungskonzept werden 2 Tage budgetiert. Die Zeit soll verwendet werden um das Konzept umzusetzen und Tests zu erstellen.
9	Multilingualität ist nichts neues und sollte daher keinen grossen Aufwand verursachen. Der Aufwand für den Einbau dieser in die Applikation wird deshalb auf einen Tag geschätzt.
10	Die Prozesslogik ist ein Kernelement der simBPMN. Um die Funktionalität sauber zu implementieren und diese ebenfalls zu testen, wird ein geschätzter Aufwand von 8 Tagen benötigt.
11	Im Unterschied von den oben genannten Tests sollen hier hauptsächlich Integrationstests an der Software vorgenommen werden. Dazu soll ein Testkonzept erstellt und verwendet werden. Damit ein stabiles Endprodukt geliefert werden kann, wird für den gesammten Aufwand mit 7 Tagen gerechnet.
12	Aus der Studienarbeit hat sich ergeben, dass administrative Angelegenheiten sehr viel Zeit in Anspruch nehmen. Die Bachelorarbeit ist in dieser Hinsicht nicht anders. Für Besprechungen sind während 17 Wochen je 2 Meetings geplant, welche jeweils 0.5 - 1 Stunde pro Meeting in Anspruch nehmen. In dieser Arbeit haben wir darum den Erfahrungswert der Studienarbeit als Richtwert genommen und diesen hoch skaliert zur Gesamtzeit der Bachelorarbeit. So entstehen 288 Stunden, welche hier eingeplant wurden.
13	Präsentationen sind ein grosser Bestandteil unserer Arbeit, um diese sauber vorbereiten und nachbereiten zu können wurden jeweils 2 Tage pro Person für je die Zwischen- sowie Abschlusspräsentation geplant. Somit kommt man aufgerechnet auf 64 Stunden oder 8 Tage.
14	Die Implementation eines Indikators für Elemente, erfordert es innerhalb vom BPMN IO Framework neue Objekte zu erzeugen. Dies ist nicht ganz trivial und wird somit mit 4 Tage geschätzt.
15	Die Ansicht von Subprozessen sollte relativ trivial sein und wurde deshalb auf einen Tag geschätzt.
16	Der Easy Mode soll nur die Basiselemente anzeigen. Dafür musst die gesamte Pallete angepasst werden.

Tabelle A.8: Arbeitspakete

A.5 Infrastruktur

- GitLab: Versionierung, Backlog
- Jira: Zeiterfassung
- LaTeX-Editor: Jedes Mitglied ist in der Wahl seines Editors frei

A.6 Qualitätsmassnahmen

Massnahme	Projekt	Beschreibung	Ziel
Gegenseitige Reviews	BA-Arbeit sowie BA-Dokumentation	Jede Zeile der Dokumentation und der Codebase wird von mindestens einem Gruppenmitglied geprüft. Bei Verbesserungsvorschlägen werden diese diskutiert und solange umgesetzt, bis beide Seiten (Autor und Prüfer) einverstanden sind. Nur abgesegnete Änderungen werden in den Master Branch comitted.	Hohe Qualität der Dokumentation und des Codes
Codestyle- und Branching Guidelines	BA-Arbeit	Umfangreiche, klar definierte und dokumentierte Guidelines zu Coding- und Branching	Vereinheitlichung des Codes und der Commit-Historie für das gesamte Repository
Testing in Build	BA-Arbeit	Umsetzen der CI/CD Practice "Make your Build Self-Testing".	Bug-freies und stabiles Produkt
Code Quality Tools	BA-Arbeit	Einsatz von Static Code Analysis Tools.	Einhalten der Code-Style Guidelines; Hohe Qualität des Codes

Tabelle A.9: Qualitätsmassnahmen

A.6.1 Code Reviews

Im Zuge jedes Merge Requests auf den Master Branch wird ein Code Review der vorgebrachten Änderungen durch mindestens ein Gruppenmitglied durchgeführt. Erst wenn das Code Review zu einem zufriedenstellendem Ergebnis gelangt, wird dem Merge Request stattgegeben.

A.6.2 Guidelines

Klar definierte Guidelines, die zu Projektbeginn definiert werden, sollen die Vereinheitlichung von Codingstyles und Branchingstrategien ermöglichen.

Dokumentation

Die Dokumentation wird mithilfe von Latex geschrieben. Für eine bessere Übersicht wird für jedes neue Thema ein separates Latex- File erstellt. Somit kann sichergestellt werden, dass zu jedem Thema nur das geschrieben wird, womit es sich beschäftigt und somit keine Verwechslungen entstehen können.

Code Style Guidelines

Als Grundlage gelten die Guidelines der jeweiligen zugrundeliegenden Sprache.

Branching- und Commit Guidelines

Diese Guidelines gelten nur für das Projekt simBPMN Visualizer.
Es wird ein Rebase Workflow verfolgt. Vor jedem Merge Request wird, falls nötig, ein Rebase des lokalen feat/bug Branches auf den Origin Master durchgeführt. Dies simplifiziert die Commit History des Repositories und verbessert damit die Übersichtlichkeit und Nachvollziehbarkeit.
Commits erfolgen regelmässig und umfassen relativ wenige Änderungen. Es wird regelmässig ins Origin Repository gepusht.

Folgende Branches werden genutzt:

Branch	Name	Beschreibung
Stable	main	Repräsentiert stabile Version des Projekts. Commits erfolgen ausschliesslich durch Development Branch.
Development	dev	Repräsentiert aktuellen Entwicklungsstand des Projekts. Commits erfolgen ausschliesslich durch Merge Requests von Feature, Bug oder Hotfix Branches.
Features/Bugfix	feat-* / bug-*	Repräsentiert Arbeit an Feature beziehungsweise Bugfix.
Hotfix	hotfix-*	Repräsentiert Hotfixes für den Stable Branch.

Tabelle A.10: Branches

Commit Message Guidelines

Commit Messages werden nachfolgendem Muster erstellt: <type>(<scope>): <subject>
Beispiel: *refactor(users-list): refactor to generic approach*

Type

feat:	Einführung eines Features
fix:	Bug fix
style:	Änderungen am Frontend
refactor:	Refaktorierung
test:	Hinzufügen von Tests

Scope

Spezifizieren des Orts der Änderung, z.B. Name der Klasse oder Methode.

Subject

Kurze und stichhaltige Beschreibung der Änderung in Englisch. In Präsens, kleingeschrieben und ohne Punkt am Ende.

Kapitel B

Mockups

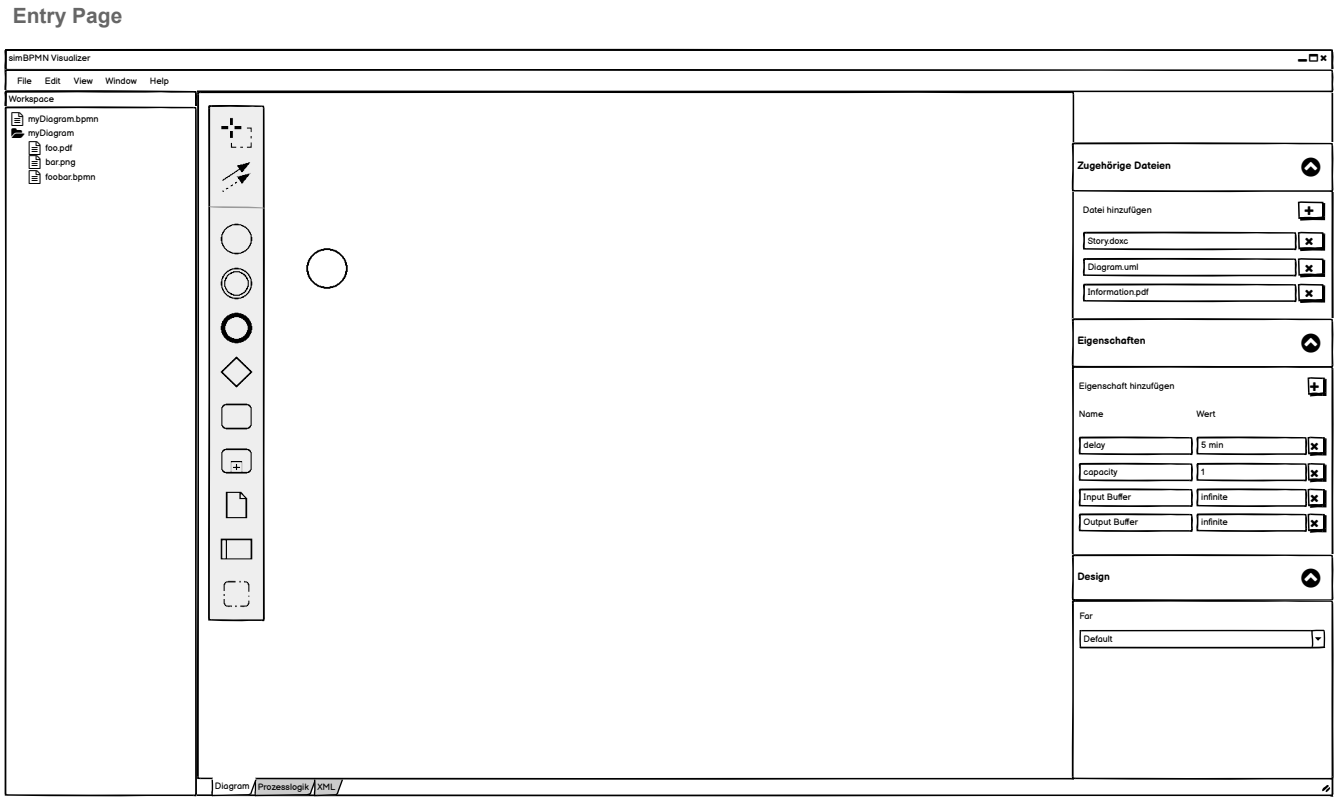


Bild B.1: Mockup Startseite

DiagramView

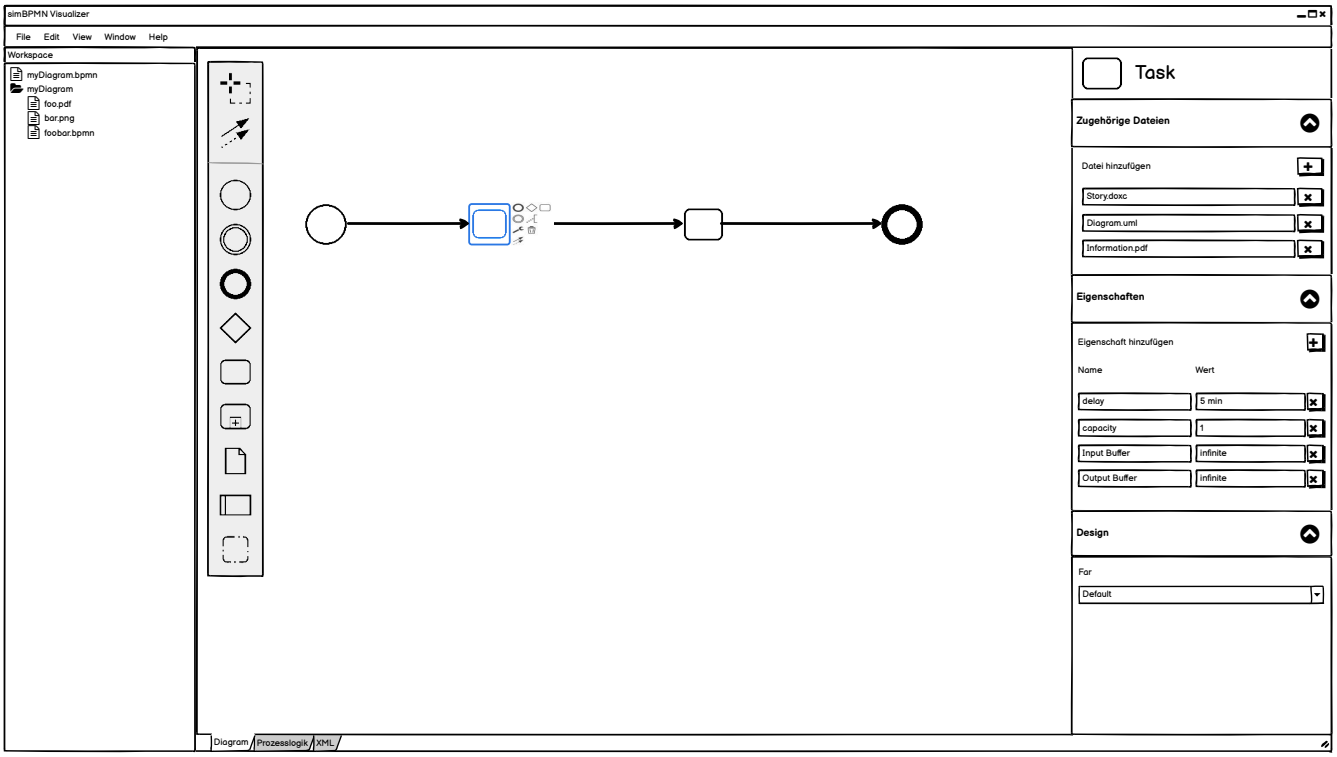


Bild B.2: Mockup Diagramansicht

Prozesslogik

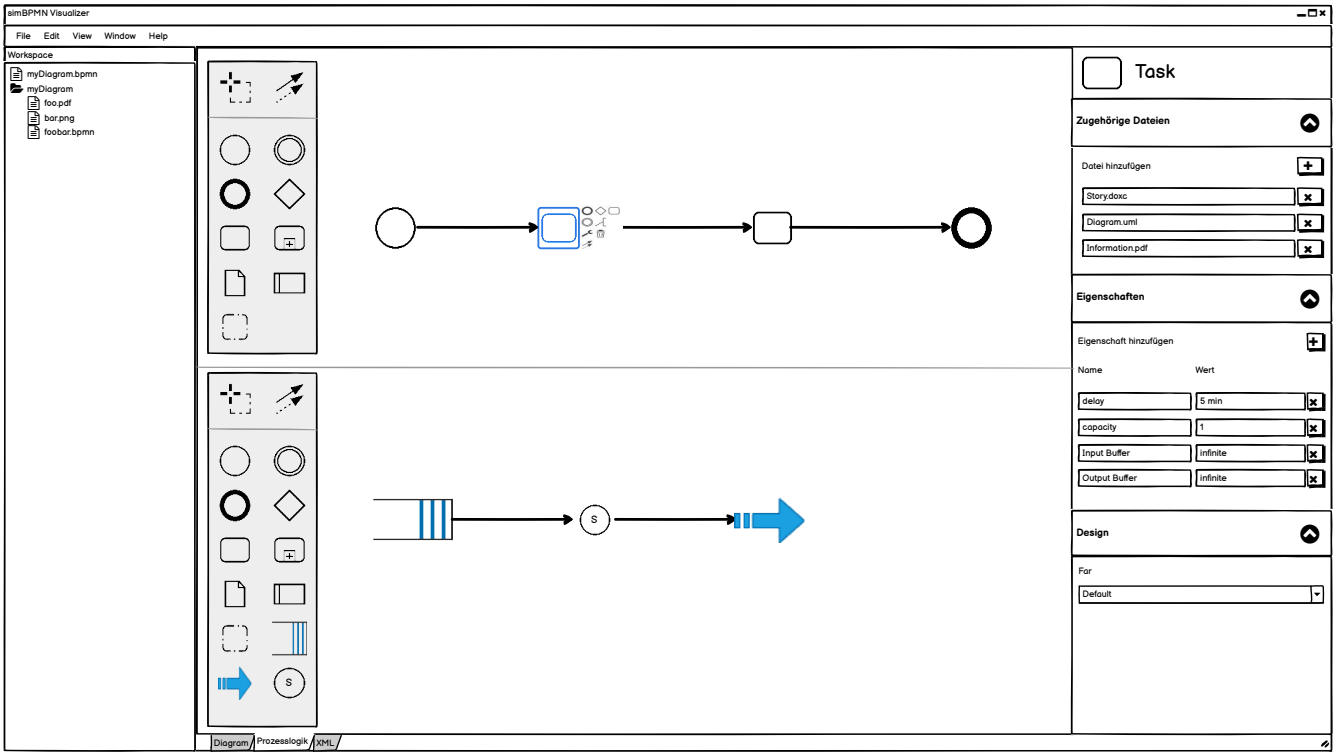


Bild B.3: Mockup Prozesslogik

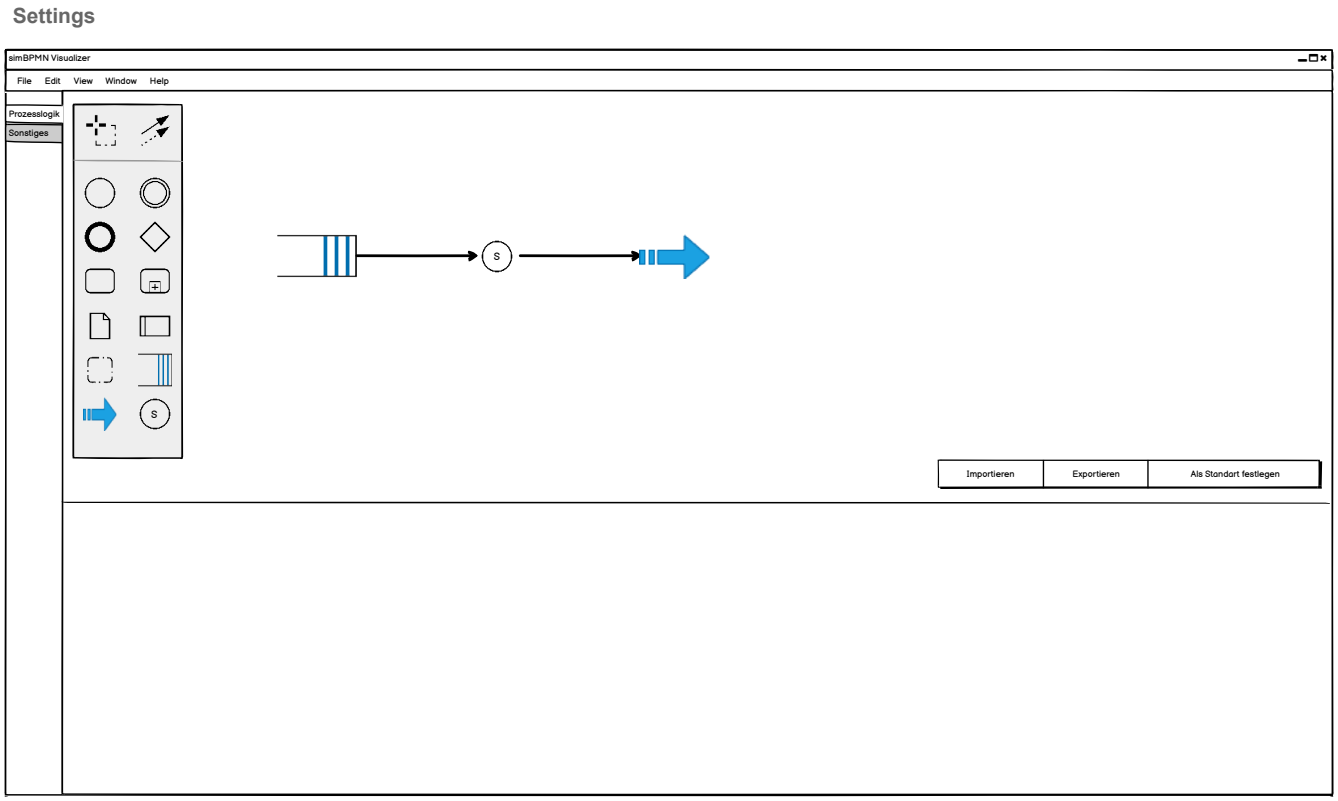


Bild B.4: Mockup Default Prozesslogik

Kapitel C

Sitzungsprotokolle

C.1 Woche 01: 01. März 2022, 13:00 Uhr

Sitzungsteilnehmer

- Prof. Dr. Andreas Rinkel
- Michel Mirsayyah
- Sven Höpfner

Traktanden

- Meetings während Ostern (12.04.2022 + 19.04.2022)
- Bekanntgabe Gegenleser
- Arbeitsplanung besprechen

Beschlüsse / Diskussionen

- Meeting am 13.04.2022 fällt aus. Meeting vom 19.04.2022 wird auf den 20.04.2022 verschoben.
- Gegenleser wird in den nächsten Wochen bekanntgegeben.
- Die Arbeit wird mithilfe von Vorveröffentlichungen geschrieben. Zu jedem Meilenstein wird diese herausgegeben und bewertet.

Nächste Aufgaben

Was	Verantwortlichkeit
Vorveröffentlichung vorbereiten	Michel Mirsayyah Sven Höpfner

C.2 Woche 02: 08. März 2022, 13:00 Uhr

Sitzungsteilnehmer

- Prof. Dr. Andreas Rinkel
- Michel Mirsayyah
- Sven Höpfner

Traktanden

- Auswahl der im Editor berücksichtigten Ereignisse und Gateways. Als Basis wird http://www.bpmb.de/images/BPMN2_0_Poster_DE.pdf verwendet
- Bestimmung der simBPMN Elemente (Warteschlange, Server)
- Bekanntgabe Gegenleser
- Status Zwischenabgabe

Beschlüsse / Diskussionen

- Die Ereignisse beschränken sich auf die Blanko Ereignisse den Timer, das Fehler- und das Abbruchereignis.
- Ereignisbasierte Gateways sind teil der Beschreibung und werden komplett mit in die simBPMN übernommen.
- Die simBPMN Elemente welche neu hinzugefügt werden, beschränken sich vorerst auf einen Eingangspuffer, eine Bearbeitungszeit und einen Ausgangspuffer.
- Gegenleser ist Ivan Bütler und wurde auf Teams bekannt gegeben
- Zwischenabgabe ist so in Ordnung, weiteres siehe unten.

Nächste Aufgaben

Was	Verantwortlichkeit
Für die erste Zwischenabgabe muss der Projektplan in den Anhang verschoben werden. Das Dokument soll anschliessend auf Teams und auch auf die Website hochgeladen werden.	Michel Mirsayyah Sven Höpfner

C.3 Woche 03: 15. März 2022, 13:00 Uhr

Sitzungsteilnehmer

- Prof. Dr. Andreas Rinkel
- Michel Mirsayyah
- Sven Höpfner

Traktanden

- Anpassung der simBPMN Elemente
- Abgabe Vorabveröffentlichung

Beschlüsse / Diskussionen

- Die BPMN Elemente welche etwas mit Transaktionen zu tun haben, werden nicht in die simBPMN übertragen
- Am Ende der Woche wird die erste Vorabveröffentlichung abgegeben

Nächste Aufgaben

Was	Verantwortlichkeit
Fertigstellung der ersten Vorabveröffentlichung	Michel Mirsayyah Sven Höpfner

C.4 Woche 04: 22. März 2022, 13:00 Uhr

Sitzungsteilnehmer

- Prof. Dr. Andreas Rinkel
- Michel Mirsayyah
- Sven Höpfner

Traktanden

- Nachbesprechung Präsentation mit Oliver Rose
- Nachfrage nach Experten für Setupprobleme am Prototypen
- Erste konzeptionelle Besprechung bezüglich der Dateispeicherung

Beschlüsse / Diskussionen

- Präsentation mit O.Rose verlief gut. Interesse scheint geweckt zu sein. Eventuell wird für das Ende der Bachelorarbeit eine Reise nach München geplant um das Tool vor Ort vorzustellen und zu besprechen.
- Das Problem wurde erklärt. Sollten wir zu keinem Resultat kommen, können wir bei Michael Gfeller oder Luc Bläser nachfragen.
- Die Dateien in der Dateispeicherung sollen alle an einem Ort liegen und dort Zugänglich sein.

Nächste Aufgaben

Was	Verantwortlichkeit
Ausarbeitung des User Interfaces	Sven Höpfner
Problemfindung am Prototypen	Michel Mirsayyah

C.5 Woche 05: 29. März 2022, 13:00 Uhr

Sitzungsteilnehmer

- Prof. Dr. Andreas Rinkel
- Sven Höpfner

Traktanden

- Status Prototyp
- Organisation der Anhänge
- Besprechung des User Interfaces
- Terminfindung Zwischenpräsentation

Beschlüsse / Diskussionen

- Die in der letzten Woche bepsrochenen Probleme konnten behoben werden und der erste Prototyp wird Planmässig fertiggestellt.
- Für die Anhänge wird der Ansatz eines Workspaces verwendet. Zusätzlich wird in die Applikation eine Übersicht über alle Dateien in diesem Workspace eingebaut.
- In der Prozesslogik sollen alle Elemente hinzugefügt werden können.
- Das Server Symbol der Prozesslogik wird um ein S erweitert um dies klar vom Task abzutrennen.
- Der Erwartungswert von der Dauer wie lange ein Server benötigt ist Pflicht
- Nach Möglichkeit soll die Prozesslogik importiert und exportiert werden können.
- Die Zwischenpräsentation soll zwischen dem 20. und 29. April stattfinden.

Nächste Aufgaben

Was	Verantwortlichkeit
Terminfindung Zwischenpräsentation	Sven Höpfner
Überarbeitung des User Interfaces	Sven Höpfner
Fertigstellung des Prototypen	Michel Mirsayyah

C.6 Woche 06: 05. April 2022, 13:00 Uhr

Sitzungsteilnehmer

- Prof. Dr. Andreas Rinkel
- Michel Mirsayyah
- Sven Höpfner

Traktanden

- Zwischenabgabge
- Zwischenpräsentation

Beschlüsse / Diskussionen

- Die nächste Zwischenabgabge wird in der Osterwoche abgegeben
- Die Zwischenpräsentation findet in am 22. April 2022 statt.
- Die nächsten Sitzungen fallen aufgrund der Ferienwoche und Ostern aus.

Nächste Aufgaben

Was	Verantwortlichkeit
Einladungen Zwischenpräsentation	Sven Höpfner
Start der Entwicklung an der Applikation	Michel Mirsayyah Sven Höpfner

C.7 Woche 07: 26. April 2022, 13:00 Uhr

Sitzungsteilnehmer

- Prof. Dr. Andreas Rinkel
- Michel Mirsayyah
- Sven Höpfner

Traktanden

- Nachbesprechung Zwischenpräsentation
- Fortschritt
- Administratives

Beschlüsse / Diskussionen

- Die Zwischenpräsentation ist gut angekommen. Die anwesenden Parteien haben positive Feedback gegenüber den Studenten geäussert
- Die Entwicklung läuft planmässig. Die Applikation wird für die nächste Woche in einen präsentablen zustand gebracht und dem Prof. Dr. Andreas Rinkel vorgestellt.
- Die Dauer der Bachelorarbeit wird um eine Woche verlängert. Der Abgabetermin bleibt jedoch der gleiche

Nächste Aufgaben

Was	Verantwortlichkeit
Applikation bereitstellen	Michel Mirsayyah Sven Höpfner

C.8 Woche 08: 03. Mai 2022, 13:00 Uhr

Sitzungsteilnehmer

- Prof. Dr. Andreas Rinkel
- Michel Mirsayyah
- Sven Höpfner

Traktanden

- Fortsschritsbesprechung
- Besprechung Validierung und Requirements

Beschlüsse / Diskussionen

- Derzeitige Funktionalität wurde vorgestellt und kam gut an.
- Es wird ein Meeting im Sommer zusammen mit der ASIM geplant um das Tool vorzustellen.
- Requirements und Validation werden unterhalb der Propertiers dargestellt
- Die Requirements und Validation erhalten jeweils einen Titel und eine Beschreibung
- Im nächsten Meeting wird ein Protoyp abgegeben, damit sich der Prof. Dr. Andreas Rinkel ein Bild von der Applikation machen kann.

Nächste Aufgaben

Was	Verantwortlichkeit
Workspace + Requirements und Validation fertigstellen	Michel Mirsayyah Sven Höpfner
Distribution erstellen	Sven Höpfner
Meeting mit der ASIM organisieren	Prof. Dr. Andreas Rinkel

C.9 Woche 09: 10. Mai 2022, 13:00 Uhr

Sitzungsteilnehmer

- Prof. Dr. Andreas Rinkel
- Michel Mirsayyah
- Sven Höpfner

Traktanden

- Abgabe Prototyp
- Fortschrittsbesprechung

Beschlüsse / Diskussionen

- Die Applikation wurde abgegeben und vorgeführt. In der nächsten Woche wird Feedback entgegengenommen und eingebaut.

Nächste Aufgaben

Was	Verantwortlichkeit
Implementation weiterführen	Michel Mirsayyah Sven Höpfner
Prototyp Austesten	Prof. Dr. Andreas Rinkel

C.10 Woche 10: 17. Mai 2022, 13:00 Uhr

Sitzungsteilnehmer

- Prof. Dr. Andreas Rinkel
- Michel Mirsayyah
- Sven Höpfner

Traktanden

- Feedback Prototyp
- Fortschrittsbesprechung

Beschlüsse / Diskussionen

- Das Feedback für den Prototyp wurde entgegengenommen und es handelt sich hauptsächlich um Darstellungsprobleme, welche bewirken, dass der Benutzer sich nicht direkt zurechtfindet.
- Die Entwicklung läuft nach dem Zeitplan und wir sind zuversichtlich die Anforderungen umsetzen zu können

Nächste Aufgaben

Was	Verantwortlichkeit
Einbau Feedback	Michel Mirsayyah Sven Höpfner

C.11 Woche 11: 24. Mai 2022, 13:00 Uhr

Sitzungsteilnehmer

- Prof. Dr. Andreas Rinkel
- Michel Mirsayyah
- Sven Höpfner

Traktanden

- Fortschrittsbesprechung

Beschlüsse / Diskussionen

- Der Fortschritt wurde besprochen, es gibt jedoch keine weiteren Handlungen für diese Woche.

Nächste Aufgaben

Was	Verantwortlichkeit
Implementation weiterführen	Michel Mirsayyah Sven Höpfner

C.12 Woche 12: 31. Mai 2022, 13:00 Uhr

Sitzungsteilnehmer

- Prof. Dr. Andreas Rinkel
- Michel Mirsayyah
- Sven Höpfner

Traktanden

- Fortschrittsbesprechung

Beschlüsse / Diskussionen

- Der Fortschritt wurde besprochen und die Applikation soll nächste Woche vorgeführt werden.

Nächste Aufgaben

Was	Verantwortlichkeit
Applikation vorbereiten	Michel Mirsayyah Sven Höpfner

C.13 Woche 13: 07. Juni 2022, 13:00 Uhr

Sitzungsteilnehmer

- Prof. Dr. Andreas Rinkel
- Michel Mirsayyah
- Sven Höpfner

Traktanden

- Besprechung der Applikation

Beschlüsse / Diskussionen

- In der Applikation können noch einige QoL Änderungen durchgeführt werden, um die Userexperience zu verbessern.
- Das Meeting in der nächsten Woche findet online statt.

Nächste Aufgaben

Was	Verantwortlichkeit
Feedback einbauen	Michel Mirsayyah Sven Höpfner

C.14 Woche 14: 14. Juni 2022, 13:00 Uhr

Sitzungsteilnehmer

- Prof. Dr. Andreas Rinkel
- Michel Mirsayyah
- Sven Höpfner

Traktanden

- Offene Punkte für die Abgabe
- Stand der Arbeit

Beschlüsse / Diskussionen

- Die Arbeit wird über das AVT abgegeben, sowie über den Teamskanal
- Am 21. Juli findet die Präsentation in München statt.

Nächste Aufgaben

Was	Verantwortlichkeit
Abgabe Vorbereiten	Michel Mirsayyah Sven Höpfner

Kapitel D

Kompendium

D.1 Setup des Projekts

Folgende Befehle werden jeweils in einem Terminal ausgeführt. Der aktuelle Pfad des Terminals bestimmt dabei den Projektpfad der Applikation.

D.1.1 Projektskelett

Mit folgendem Befehlt wird ein Skelett für eine Electron-Applikation mit Hilfe von Electron Forge erzeugt. Hier wird diese unter dem Namen "my-new-app" erstellt.

```
npx create-electron-app my-new-app --template=typescript-webpack
```

Die Projektstruktur wird nach einer kurzen Zeit erstellt und sollte wie folgt aussehen:

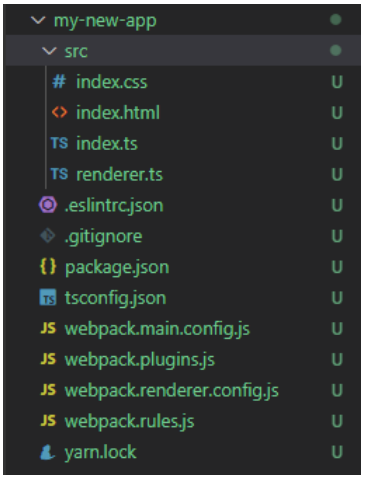


Bild D.1: Electron Projektskelett

Um mit Electron zu interagieren muss mit weiteren Console-Commands gearbeitet werden. Da hier prinzipiell eine Node JS Applikation gebaut wurde, könnte hier mittels NPM oder YARN gearbeitet werden. NPM wird für unsere Zwecke leider entweder nur Teilweise oder gar nicht unterstützt.

```
# dependencies herunterladen
yarn

# electron starten
yarn start

# electron als Applikation builden
yarn make
```

Um zu testen was erzeugt wurde kann mit obig gezeigten Befehlen Electron gestartet werden. Zuerst sollten alle Dependencies mit "yarn" heruntergeladen werden. Anschliessend kann Electron mit dem Start-Befehl ein Userinterface öffnen, dieses sollte wie folgt aussehen:

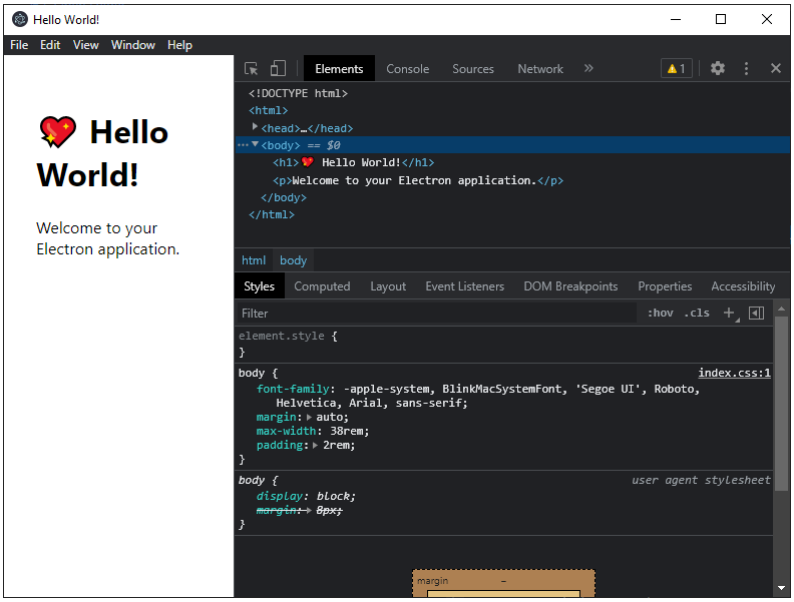


Bild D.2: Electron UI

Mithilfe von "yarn make" kann die Applikation bereits für die verschiedenen Systeme erstellt werden, wodurch man die verschiedenen Dateien (.exe, .dmg, usw...) im entsprechend definierten Ordner findet.

D.1.2 Test-Framework

Als nächstes muss ein Test-Framework hinzugefügt werden, damit der Code automatisch geprüft werden kann. Dazu können wir wiederum aus einer grossen Menge an Unit-Test Frameworks auswählen, die für Node JS existieren. In diesem Beispiel wird das Jasmine Framework verwendet, dieses kann wie folgt installiert werden:

```
# jasmine dem Projekt hinzufuegen
yarn add jasmine

# jasmine initialisieren
# erzeugt config file sowie Testordner
npx jasmine init
```

Nach erfolgreicher Installation sollte ein neuer Ordner namens "spec" im Projekt erscheinen, worin sich die Konfigurationsdatei von Jasmine befindet. In diesem Ordner können nun Tests definiert werden die das Framework behandelt. Wichtig dabei ist das alle Testfiles das Suffix ".spec.js" besitzen müssen.

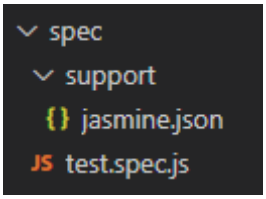


Bild D.3: Jasmine Ordnerstruktur

Um den Testvorgang zu vereinfachen, kann im File "package.json" ein weiteres Script definiert werden. Dieses dient als Alias für den formulierten Befehl.

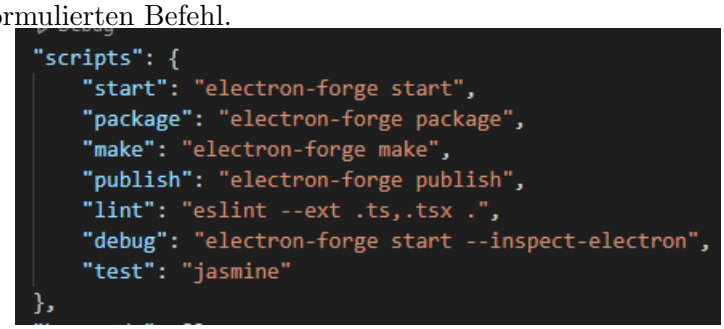


Bild D.4: Jasmine Ordnerstruktur

Mit der obigen Definition kann wie folgt getestet werden:

```
yarn test
```

D.1.3 BPMN-JS

Das BPMN-JS Framework kann auf die gleiche Weise wie obig schon erklärt hinzugefügt werden.

```
yarn add bpmn-js
yarn add bpmn-js-properties-panel
```

Die Libraries sind nun hinzugefügt und können verwendet werden.

D.2 Übersicht

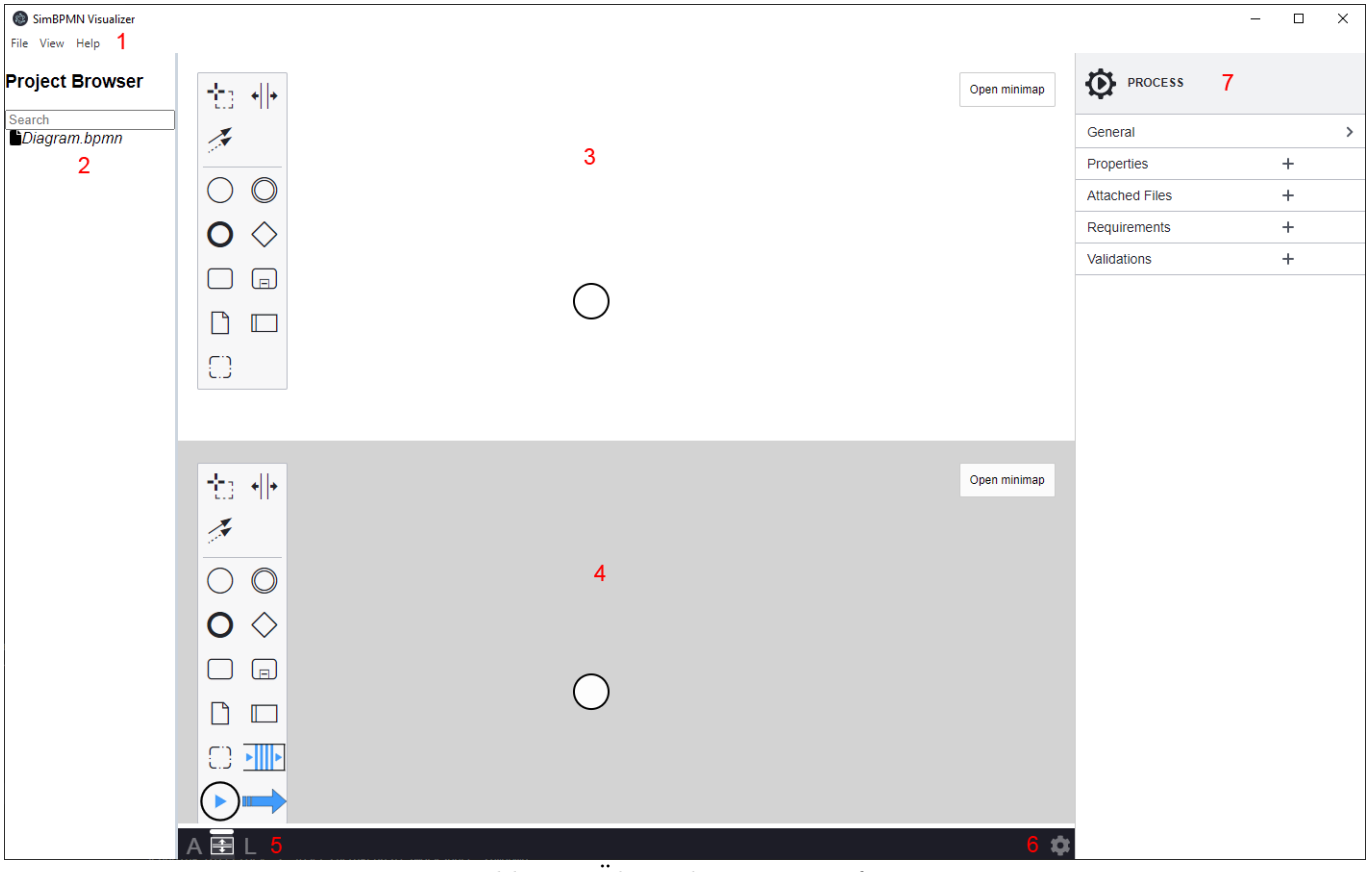


Bild D.5: Übersicht User Interface

1. Electron Menü
2. Workspace
3. Architektur BPMN
4. Logik BPMN
5. Tabswitcher
6. Einstellungen
7. Property-Panel

D.3 Electron

Der Aufbau von Electron kann in zwei Teile aufgeteilt werden, einerseits den Aufbau von Electron selbst und andererseits dem Renderer. Innerhalb des Aufbaus von Electron wird beispielsweise definiert wie gross das angezeigte Fenster sein soll. Im Renderer hingegen werden alle Elemente referenziert die inhaltlich eine Abhängigkeit zur Node JS Applikation besitzen. Alle benötigten CSS, JS, und TS Files werden deshalb im RENDERER.TS referenziert. Technisch gesprochen werden die Skripte des Renderers erst geladen nachdem

das Fenster erstellt wurde, somit können keine JS-Files ausgeführt werden die im INDEX.TS referenziert werden.

D.4 Prozessmodell

D.4.1 Hintergrundinformation

Electron basiert auf Chromium und ist daher wie ein Webbrowser aufgebaut. Moderne Webbrowser verwenden heutzutage unabhängige Prozesse um ihre Webseiten anzuzeigen. Dies führt zwar zu mehr Aufwand um jeden Tab zu verwalten, allerdings bietet es den Vorteil, dass wenn die Webseite abstürzt oder hängen bleibt, der Browser davon nicht beeinträchtigt wird.

D.4.2 Hauptprozess

Jede Electron Applikation besitzt genau einen Hauptprozess. Dieser wird als Einstiegspunkt in die Applikation verwendet. Zusätzlich läuft dieser in einer Node.js Umgebung und besitzt somit die Möglichkeit Module und das gesamte Node.js API zu verwenden.

D.4.3 Renderprozess

Für jedes offene Fenster startet Electron einen neuen Renderprozess. Der Prozess ist dafür zuständig die Webelement der entsprechenden Webseite zu rendern. Der Renderprozess besitzt jedoch im Gegensatz zum Hauptprozess nicht die Möglichkeit Module oder das Node.js API zu verwenden.

D.4.4 Kommunaktion zwischen den Prozessen

Um eine Kommunikation zwischen den Prozessen zu ermöglichen, können Nachrichten durch definierte Kanäle in der preload Datei gesendet werden. Ein Beispiel dazu wird wie folgt definiert.

Preload

In der preload Datei wird Kanal für die entsprechenden Nachrichten definiert.

preload.js

```
const { contextBridge, ipcRenderer } = require('electron')

contextBridge.exposeInMainWorld('electronAPI',{
    openFile: () => ipcRenderer.invoke('dialog:openFile')
})
```

D.5 Multilingualität

Wie auch schon beim Setup muss hier wieder zwischen Electron und Applikation unterschieden werden. Es müssen beide wie folgt verändert werden um Multilingualität zu unterstützen.

D.5.1 BPMN JS

Im File "regularBPMN.js" ist definiert wie das BPMN Framework eingesetzt wird. Dieses kann modular erweitert werden. Hier wurde ein neues Objekt erstellt, in welchem die Übersetzungen zu finden sind, und dem BPMN-Modeler hinzugefügt.

app.js

```
import bpmnTranslations from '../translations/bpmn/translations';

var customTranslateModule = {
  translate: ['value', bpmnTranslations]
};
```

```
var bpmnModeler = new BpmnModeler({
   container: canvas,
   propertiesPanel: {
      parent: '#js-properties-panel'
   },
   additionalModules: [
   BpmnPropertiesPanelModule,
   BpmnPropertiesProviderModule,
   customTranslateModule
   ]
});
```

translations.js

```
import translations from './bpmntranslations';

export default function customTranslate(template, replacements) {
   replacements = replacements || {};

   // Translate
   template = translations[template] || template;

   // Replace
   return template.replace(/{([^}]+)}/g, function (_, key) {
   return replacements[key] || '{' + key + '}';
});
}
```

bpmntranslations.js

```
export default {
   'Exclusive Gateway': 'Exklusives Gateway',
   'Parallel Gateway': 'Paralleles Gateway',
   'Inclusive Gateway': 'Inklusives Gateway',
   'Complex Gateway': 'Komplexes Gateway',
   ...
}
```

Beim starten der Applikation werden die entsprechenden Texte aus dem hinzugefügten Objekt übernommen.

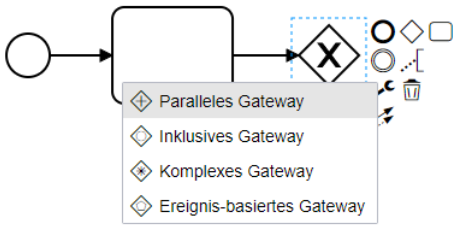


Bild D.6: Übersetzung BPMN JS

D.5.2 Electron

Für die Anpassung der Multilingualität in Electron muss das File INDEX.TS angepasst werden siehe D.3 Anpassung Electron. Hier wurde für die Mehrsprachigkeit das Framework I18Next verwendet. Dieses muss wie folgt konfiguriert werden. Der Befehl "changeLanguage(...)" steuert dabei die Ausgabesprache und kann von einem beliebigen Ort aufgerufen werden.

index.ts

```
const i18n = require('./configs/i18next.config');
const config = require('./configs/app.config');
```

```
const menuFactoryService = require('./menus/menuFactory');

i18n.on('loaded', (loaded: any) => {
   i18n.changeLanguage('en');
   i18n.off('loaded');
});

i18n.on('languageChanged', (lng: any) => {
   menuFactoryService.buildMenu(app, mainWindow, i18n);
   mainWindow.webContents.send('language-changed', {
      language: lng,
      namespace: config.namespace,
      resource: i18n.getResourceBundle(lng, config.namespace)
   });
});
```

Damit das Programm weis welche Sprachen existieren und wo die übersetzten Texte zu finden sind, müssen Konfigurationsdateien erstellt werden. Diese können wie folgt aussehen:

app.config.js

```
module.exports = {
   platform: process.platform,
   port: process.env.PORT ? process.env.PORT : 3000,
   title: 'simBPMN',
   languages: ['en', 'de'],
   fallbackLng: 'en',
   namespace: 'translation'
};
```

i18next.config.js

```
const i18n = require('i18next');
const i18nextBackend = require('i18next-node-fs-backend');
const config = require('../configs/app.config');

const i18nextOptions = {
   backend: {
      loadPath: './src/translations/electron/{{lng}}.json',
      jsonIndent: 2,
   },
   interpolation: {
      escapeValue: false
   },
   saveMissing: true,
   fallbackLng: config.fallbackLng,
   whitelist: config.languages,
   react: {
      wait: false
   }
};

i18n
.use(i18nextBackend);

if (!i18n.isInitialized) {
   i18n
   .init(i18nextOptions);
}

module.exports = i18n;
```

Übersetzungen werden in einem JSON-Format erstellt welches aus Key-Value Paaren besteht. Anhand des

Keys und des Sprachcodes (bsp. en = Englisch, de = Deutsch) werden die eingetragenen Werte dargestellt. Hier folgend ist ein Beispiel zur Englischen Übersetzung:

en.json

```
{
   "Quit": "Quit",
   "View": "View",
   "Reload": "Reload",
   "Full Screen": "Full Screen",
   "Minimize": "Minimize",
   "Toggle Developer Tools": "Toggle Developer Tools",
   "Help": "Help",
   "About App": "About App",
   "de": "German",
   "en": "English",
   "Language": "Language"
}
```

Die angebotene Funktion des Übersetzungsframework kann wie folgt aufgerufen werden:

```
//<variable> = i18n.t(<Key>)
label = i18n.t('Reload')
```

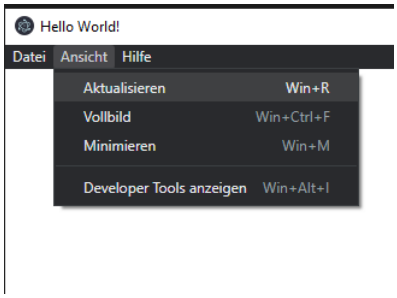


Bild D.7: Übersetzung in Electron

D.6 Electron Menu anpassen

Electron bietet eine eigene Klasse für Anpassungen im Menü. Dieser kann ein Template übergeben werden worin definiert ist wie das Menü aussehen soll. Als Beispiel folgt ein Ausschnitt aus unserer Menüdefinition:

menu.js

```
import { Menu } from "electron";

const template = [{
   label: i18n.t('File'), //get translation for File
   submenu: [
   {
      label: i18n.t('CreateFile'),
      accelerator: 'CommandOrControl+O',
      click: function () {
         ...
      }
   },
   {
      label: i18n.t('SaveFile'), //get translation for SaveFile
      accelerator: 'CommandOrControl+S',
      click: async function () {
         ...
      }
   },
   {
```

```
        type: 'separator'
    },
    .
    .
    .
}];


const menu = Menu.buildFromTemplate(template);
Menu.setApplicationMenu(menu);
```

Weitere Informationen sind über die offiziellen Electron Docs zu finden.

D.7 Sidebar

Die Sidebar, hier am linken Rand des User-Interfaces zu erkennen, ist dazu da dem Benutzer eine Hilfe zu bieten, damit dieser sich in seinem Projekt-Workspace zurecht finden kann. In der Anzeige werden alle Inhalte des Workspaces dargestellt. Um präziser zu sein sind alle Verknüpften Dateien des Projekts darin enthalten. Wenn also dem Benutzer nicht mehr geläufig ist auf welchem Objekt er eine gewisse Datei hinterlegt hat, dann kann dieser mittels der Suchfunktion die gewünschte Datei finden. Diese lässt sich dann mittels eines Klicks öffnen.

D.7.1 Implementation

Die Sidebar hat nichts mit dem BPMN IO Framework zu tun und ist eine reine Typescript Anwendung mit Electron. Es kann deshalb als separate Funktion betrachtet werden.

D.7.2 Interprocess Communication

Innerhalb von einer Electron Anwenung muss immer Unterschieden werden in welchem Kontext etwas laufen muss. Um dies ein wenig klarer zu machen verfolgen wir das folgende hypothetische Beispiel. Wir möchten aus dem Menü unter dem Menüpunkt "Datei" und "Datei öffnen" der Applikation ein einfaches Text-File von unserer Maschine auswählen und dessen Inhalt in unserer Electron App anzeigen. Um dies zu bewerkstelligen müssen wir nun entscheiden ob unsere Skripte im Main Prozess oder im Renderer Prozess der Applikation laufen sollen. Innerhalb des Main Prozesses müssen alle Funktionen ausgeführt werden, die mit der Anwendermaschine interagieren, wie beispielsweise unser FileDialog den wir benötigen um ein File auszuwählen. Alles was mit unserer Webapp zu tun hat muss im Renderer Prozess ausgeführt werden. Jetzt haben wir ein Dilemma, wir müssen nämlich den Code für das Menü im Renderer und die Klick-Funktion des Menüeintrags im Main Prozess laufen lassen. Wenn der Code einfach "normal" eingebettet wird, wird alles im Renderer ausgeführt und wir erhalten eine Fehlermeldung. Wir müssen also irgendwie definieren können welcher Code in welchem Kontext ausgeführt werden soll. Hier kommt die Interprocess Communication ins Spiel. Damit wird es uns ermöglicht Funktionen in einem anderen Kontext auszuführen.

Von Main zu Renderer wechseln

Folgende Definitionen müssen in den oberen beiden Files erstellt werden um diese dann anschliessend im letzten File anwenden zu können.

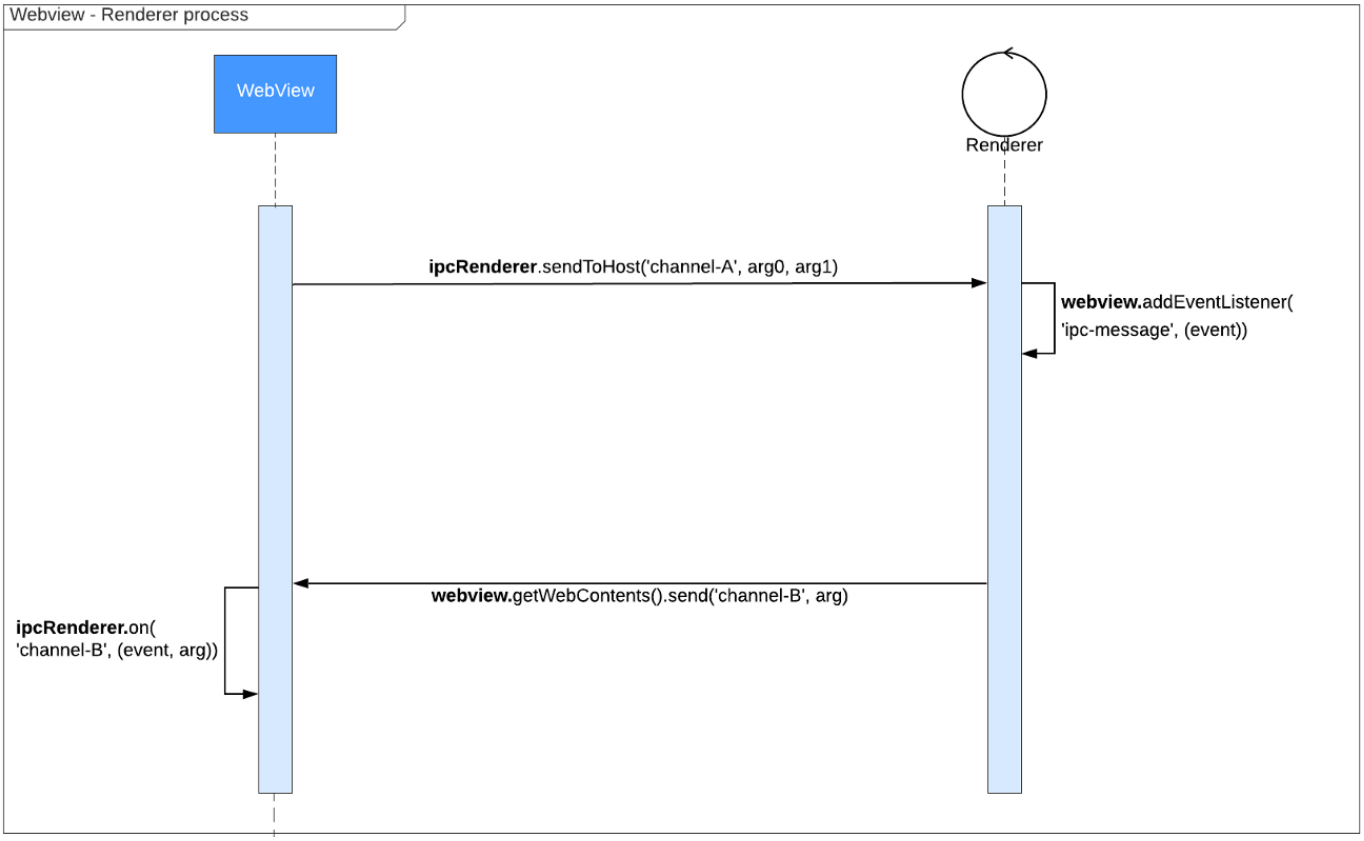


Bild D.8: WebView - Main
[3]

preload.ts

```
import { ipcRenderer, contextBridge } from 'electron';

contextBridge.exposeInMainWorld('electronAPI', {
   openTextFile: (callback: any) => ipcRenderer.on("openTextFile", callback)
});
```

renderer.ts

```
(window as any).electronAPI.openTextFile((event: any, value: any) => {
   ... Code to display file content
});
```

menu.js

```
mainWindow.webContents.send("openTextFile", filepath);
```

Von Renderer zu Main wechseln

Hier wird die Kommunikation von der anderen Seite aus beschrieben. Gehen wir hier davon aus dass wir einen Dateipfad besitzen und herausfinden möchten ob es sich bei diesem um ein Verzeichnis oder File handelt.

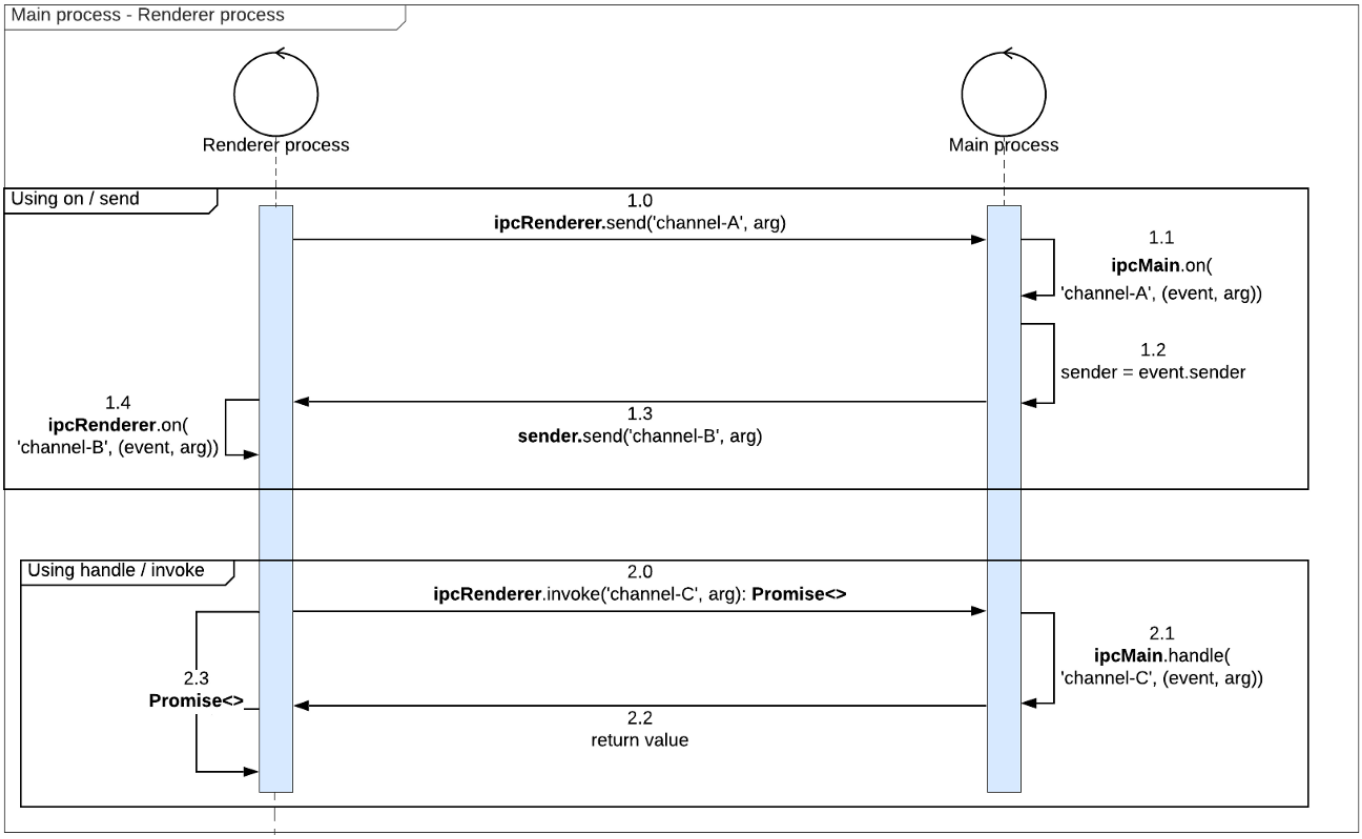


Bild D.9: Main - Renderer
[3]

preload.d.ts

```
export interface IElectronAPI {
   isDirectory: (filePath: string) => Promise<boolean>
}

declare global {
   interface Window {
      electronAPI: IElectronAPI
   }
}
```

preload.ts

```
import { ipcRenderer, contextBridge } from 'electron';

contextBridge.exposeInMainWorld('electronAPI', {
   openTextFile: (callback: any) => ipcRenderer.on("openTextFile", callback),
   isDirectory: (filePath: string) => ipcRenderer.invoke("isDirectory", filePath)
});
```

index.ts

```
   ipcMain.handle("isDirectory", (event: any, path: string) => {
      const stats = fs.statSync(path);
      return !stats.isFile();
   });
```

File_Im_Renderer.ts

```
var result = await (window as any).electronAPI.isDirectory(path);
```

D.7.3 Suchfunktion & öffnen von Dateien

Beide Funktionen wurden mit dem Javascript Document Object gelöst. Die Suchfunktion ist eifache CSS Magie in welcher einfach alle Elemente ausgeblendet werden die nicht dem Suchkriterium entsprechen. Das Öffnen von Dateien wird beim erstellen der HTML Objekte als Klick-Event mit entsprechendem Pfad hinzugefügt.

D.8 Tab Switcher

Der Tab Switcher ermöglicht es dem Benutzer zwischen verschiedenen Ansichten umzuschalten um effizienter Arbeiten zu können. Dazu steht es dem Benutzer frei zwischen der BPMN Vollansicht, der Split View und der simBPMN Vollansicht umzuschalten.
Auch hier wurde wieder mit CSS Magie gearbeitet um nur die gewünschten Inhalte dem Benutzer darzustellen.

D.8.1 Übernahme von Code

Das grafische Element für den Tab Switcher wurde inspiriert von einer Vorlage auf CodePen[4]. Dieses wurde von dort übernommen und entsprechend für unsere Zwecke modifiziert. Alle Vorlagen von Codepen.io unterliegen der MIT-Lizenz und dürfen weiterverwendet werden. Das übernommene Element ist vom Ersteller Sikriti Dakua[5][6] und wurde ebenfalls gemäss der MIT-Lizenz im Code so gekennzeichnet.

D.8.2 Übersicht

Bild D.10: Tab Switcher

Anwender der Applikation können auf der linken Seite die Ansicht zwischen Architektur, Split-View und Logik ändern. Mit der Schaltfläche auf der rechten Seite können die Applikationseinstellungen angezeigt werden.

D.9 BPMN IO Cores

Das BPMN IO Framework wird in dieser Applikation an zwei Instanzen verwendet. Einerseits als "normale" Oberfläche zur Interaktion des Benutzers für BPMN Prozesse und andererseits als erweiterte Oberfläche für simBPMN. Mit BPMN IO Core ist jeweils eine Instanz davon gemeint.

D.9.1 Aufbau

Der BPMN Core beinhaltet den mitgelieferten Umfang des Frameworks. Zusätzlich dazu ist dort das Property-Fenster mit den entsprechenden Funktionen zum Anhängen bzw. weiter definieren eines Objekts definiert. Im simBPMN Core wurde die Basis des Frameworks mit den simBPMN Elementen erweitert. Dieser Core besitzt keine Abhängigkeit zum BPMN Core.

D.9.2 Code

Das BPMN IO Framework benötigt einen eigenständigen DIV im HTML, um korrekt angezeigt werden zu können. Im HTML Dokument muss daher ein DIV bereitgestellt und mit einer ID versehen werden. Die BPMN IO Elemente, wie zum Beispiel der Canvas, werden dann zur Laufzeit des Programms in den entsprechenden DIV mit der definierten ID geladen.

Im Backend wird definiert welche Module im BPMN Container werden und wie dieses konfiguriert sind. Um dies ein wenig einfacher zu erklären wird hier folgend ein Ausschnitt des simBPMN.js Dokumentes dargestellt:

simBPMN.js

```
var bpmnModeler = new BpmnModeler({
   container: canvas, /*canvas = $("#js-simbpmncanvas")*/
```

```
   additionalModules: [ /*Verwendete Module*/
   minimapModule,
   BpmnPropertiesPanelModule,
   ControlsModule,
   SimBPMNControlsModule,
   BpmnColorPickerModule,
   ExtensionPropertiesProvider,
   customTranslateModule
   ],
   moddleExtensions: {
      simbpmn: simBpmnModdleDescriptor, /*definition der Objekte / Erweiterungen */
   },
});
```

D.9.3 Kommunikation zwischen den Cores

Um zu einen späteren Zeitpunkt Validierungen und andere Logik effizient implementieren zu können, muss es möglich sein zwischen den Cores kommunizieren zu können. In unserem Fall konkret wurde dies bereits benötigt, um die Logik der BPMN Elemente zu laden bzw. zu speichern.
Für den Austausch zwischen den Cores wird eine asynchrone, eventbasierte Kommunikation angestrebt. Beide implementierten Cores werden im Renderer Context von Electron ausgeführt. Eine direkte Kommunikation von Renderer zu Renderer ist gemäss Electron nicht möglich, zumindest zum Zeitpunkt dieser Arbeit. Um dieses Problem zu umgehen werden die Events über ein Relay im Main Prozess weitergeleitet. Dies hat sicher noch Verbesserungspotenzial in der Zukunft, ist aber gemäss Electron die vorgeschlagene Lösung.