NOVIA
UNIVERSITY OF APPLIED SCIENCES

Modeling & Designing Toolkit For BPMN Processes

Toolkit for creating a visual representation of BPMN workflows.

Tobias Lyttbacka

Bachelor's Thesis for Bachelor of Engineering

Degree Programme in Electrical Engineering and Automation
Vaasa 2022

NOVIA
UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS

Author: Tobias Lyttbacka

Degree Programme and place of study: Electrical Engineering and Automation, Vaasa

Specialisation: Information Technology

Supervisor(s): Kaj Wikman, Vesa Mustonen, Jarkko Pukkila

Title: Modeling & Designing Toolkit For BPMN Processes

Date: 8.3.2022 Number of pages: 41

Abstract

Business Process Modeling and Notation (BPMN) has been on the rise over the past few years with no indication to slow down, the growth peaking at an adaptation rate of 60% in 2011 within corporations that use process modeling. The objective of BPMN is to provide a standard notation that can be understood by all involved parties, due to it being remarkably comprehensive, it has led many corporations including Wärtsilä to adopt the usage of such technologies. As some of the technologies provided by other companies did not meet the requirements of the organization, it was decided that said technologies would be internally developed instead.

This thesis aims to provide a comprehensive understanding of how the creation of BPMN toolsets was accomplished and how solutions were found to problems faced within Wärtsilä when attempting to further support the usage of BPMN inside its organization.

The result of this thesis was the creation of an application that would allow Wärtsilä to create, host, and manage business processes within its organization, all neatly embedded in one web application. Some of the issues solved by this application were process-versioning and collaboration between users.

Language: English
Key Words: Business Process Modeling and Notation, process modeling, business processes

NOVIA
UNIVERSITY OF APPLIED SCIENCES

EXAMENSARBETE

Författare: Tobias Lyttbacka
Utbildning och ort: El- och Automationsteknik, Vasa
Inriktning: Informationsteknik
Handledare: Kaj Wikman, Vesa Mustonen, Jarkko Pukkila

Titel: Modellerings- och designverktyg för BPMN-processer

Datum: 8.3.2022 Sidantal: 41

Abstrakt

Business Process Modeling and Notation (BPMN) har varit på uppgång med en stor ökning av användning under de senaste åren, med den största ökning under 2011 med en användning på 60 % inom företag som modeller affärsprocesser. BPMN vill möjliggöra användningen av ett standardiserat notationsspråk som skall kunna anses vara lättfattligt mellan företag, branscher och yrkesgrupper och tack vare dess lättförståelighet har många företag samt Wärtsilä valt att använda dessa teknologier. Eftersom produkter som redan finns på marknaden inte uppfyllt de kraven som ställs av Wärtsilä, beslöts det att dessa teknologier skulle utvecklas inom Wärtsilä.

Målet med detta examensarbete var att ge en förståelse för hur utveckling av dessa BPMN-relaterade verktyg möjliggjordes samt lösningen av de problem som uppstod för Wärtsilä när de försökte vidare stödja användningen av BPMN inom företaget.

Examensarbetet resulterade i en applikation som skall möjliggöra skapandet, sparandet och hanteringen av affärsprocesser för företaget, allt inbäddat i en och samma webbapplikation. Några av de problem som löstes av applikationen var till exempel, processversionshantering och möjligheten att samarbeta mellan kollegor.

Språk: engelska
Nyckelord: affärsprocessmodelleringsnotation, affärsprocesser modellering, affärsprocesser

NOVIA
UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ

Tekijä: Tobias Lyttbacka
Koulutus ja paikkakunta: Sähkö- ja automaatiotekniikka, Vaasa
Suuntautumisvaihtoehto: Tietotekniikka
Ohjaajat: Kaj Wikman, Vesa Mustonen, Jarkko Pukkila

Nimike: Mallintamisen ja suunnittelun työkalusarja BPMN-prosesseille

Päivämäärä: 8.3.2022 Sivumäärä: 41

Tiivistelmä

Business Process Modeling and Notation (BPMN) on ollut nousussa muutaman viime vuoden ajan, eikä ole viitteitä sen hidastumisesta. Kasvu saavutti 60 %:n mukautumisnopeuden vuonna 2011 prosessimallinnusta käyttävissä yrityksissä.

BPMN:n tavoitteena on tarjota standardoidun merkintäkielen käytön, jonka kaikki osapuolet ymmärtävät, koska se on erittäin kattava, ja se on saanut monet yritykset, mukaan lukien Wärtsilä, ottamaan käyttöön tällaisia teknologioita.

Koska osa muiden yritysten tarjoamista teknologioista ei vastannut organisaation vaatimuksia, päätettiin, että mainitut tekniikat kehitettäisiin sen sijaan sisäisesti Wärtsilässä.

Tämän opinnäytetyön tavoitteena oli antaa kattava käsitys siitä, kuinka BPMN-työkalusarjojen luominen onnistui ja kuinka Wärtsilän ongelmiin löydettiin ratkaisuja yrittäessään tukea BPMN:n käyttöä organisaatiossaan.

Tämän opinnäytetyön tuloksena syntyi sovellus, jonka avulla Wärtsilä voi luoda, isännöidä ja hallita liiketoimintaprosesseja organisaatiossaan, kaikki siististi upotettuna yhteen verkkosovellukseen. Joitakin tämän sovelluksen ratkaisemista ongelmista olivat prosessien versiointi ja käyttäjien välinen yhteistyö.

Kieli: englanti
Avainsanat: liiketoimintaprosessien mallinnusmerkintä, liiketoimintaprosessien mallinnus, liiketoimintaprosessit

NOVIA
UNIVERSITY OF APPLIED SCIENCES

Table of Contents

NOVIA
UNIVERSITY OF APPLIED SCIENCES

Dictionary

Abbreviation	Meaning	Page
API	Application Programming Interface	9
BPMN	Business Process Management and Notation	1
BPMI	Business Process Management Initiative	2
DMN	Decision Model and Notation	15
NPM	Node Package Manager	19
SPA	Single-Page Application	19
XML	Extensible Markup Language	8

NOVIA
UNIVERSITY OF APPLIED SCIENCES

Acknowledgments

I am very grateful to have been able to take part in the development of this project because it has been such a great learning experience for me, I want to give special thanks to the managers Vesa Mustonen, Nishant Redekar, Jarkko Pukkila, and Henri Salmi for giving me this opportunity.

NOVIA
UNIVERSITY OF APPLIED SCIENCES

1 Introduction

From a small sawmill in 1834 to a frontier for engineering innovation, while operating in over 200 locations in 68 different countries Wärtsilä is a global leader in the marine and energy markets with its innovative technologies and lifecycle solutions. Wärtsilä is constantly pushing toward a low carbon feature by providing the world with services and solutions that maximize the economic and environmental performance of vessels, power plants, and entire systems. Wärtsilä plays an important part in supplying the world with sustainable energy. (Wärtsilä, n.d.)

1.1 Purpose

This thesis aims to give a comprehensive understanding of development regarding BPMN-related applications, and how different issues faced within Wärtsilä related to such developments were overcome. As solutions provided by other companies did not meet the requirements needed for Wärtsilä it was decided that the solution should be internally developed instead, some of the unfulfilled requirements were associated with sustainability and scalability.

1.2 Solution

This thesis will mainly focus on Design-Studio, which is an internally developed BPMN application, which was developed to further support the usage of BPMN within Wärtsilä's organization. Design-Studio was designed to allow for sustainable process mapping, which was done through supporting sustainable features such as process -versioning and variance. This, allows Wärtsilä to create, host, and manage its business process in an organized manner, which allows Wärtsilä to gain the full advantage of BPMN.

NOVIA
UNIVERSITY OF APPLIED SCIENCES

2 BPMN Introduction

Business Process Modeling Notation (BPMN) is a visual modeling language for business analysis applications, it utilizes a method of visualizing planned business processes in an open standard notation for graphical flowcharts that can easily be understood by business stakeholders, business analysts, software developers, and data architects. (Visual Paradigm, n.d.)

2.1 History

BPMN was originally developed and published in 2004 by the Business Process Management Initiative (BPMI), but since the merging of Object Management Group (OMG) and BPMI in 2005 it has been maintained by OMG. In 2010 version 2.0 was developed, and it uses a broader set of symbols and notations to create a more detailed rule for Business Process Diagrams. (Lynch, 2022; Visual Paradigm, n.d.)

2.2 Benefits

BPMN's graphical presentation provides a standardized common language, which is much easier to understand than narrative text therefore can all involved parties whether technical or non-technical be given significantly more sufficient details as to how the process should be implemented, ideally it would bridge the gap between the implementation and the process intention, hence achieving the main purpose of BPMN which is to increase operational efficiency while also allowing for easier communication and collaboration within an organization. (Lynch, 2022; Visual Paradigm, n.d.)

2.2.1 High Level

On a high level, BPMN will aid participants and other stakeholders to gain a more throughout understanding of the process through its easy-to-understand visual representation of the steps involved. (Lucidchart, 2022)

NOVIA
UNIVERSITY OF APPLIED SCIENCES

2.2.2 Low Level

People at a more involved level are given the clarity of the sequence of business activities, BPMN also gives sufficient information as to how the process should precisely be implemented. (Lucidchart, 2022)

NOVIA
UNIVERSITY OF APPLIED SCIENCES

3 BPMN Notation Symbols

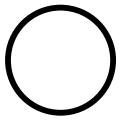
As of BPMN version 2.0, workflows consist of four different categories to visually represent a business process which are Flow Objects, Connecting Objects, Swimlanes, and Artifacts. (Lucidchart, 2022; Lynch, 2022; Wondershare EdrawMax, 2022)

3.1 Flow Objects

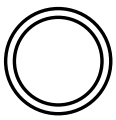
The flow objects are the main elements used to describe the workflow within BPMN, consisting of three different core elements which are Events, Activities, and Gateways. (Visual Paradigm, n.d.)

3.1.1 Events

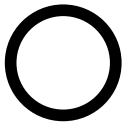
Events are represented by a circle and can include a symbol based on the event type, every BPMN workflow should at least include a start and end event.



Start Event
Start events represent the first step in a business process and are therefore required because every BPMN workflow needs to have an initiating event. (Lucidchart, 2022)



Intermediate Event
Events that occur between start and end events, the direction of the connection is determining whether the event is catching or throwing information, in other terms whether it is delivering or receiving data.
(Lucidchart, 2022)

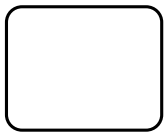


End Event
End events are recognized by their thick black line and represent the final step of a BPMN workflow. (Lucidchart, 2022)

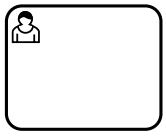
NOVIA
UNIVERSITY OF APPLIED SCIENCES

3.1.2 Activities

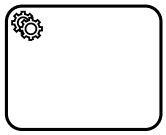
Activities are used to represent the different kinds of work being done within a BPMN workflow. (Lucidchart, 2022)



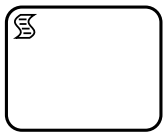
Task
The basic task is used to represent the most simplistic activity at such a level that it is not possible to simplify an assignment any further. (Wondershare EdrawMax, 2022)



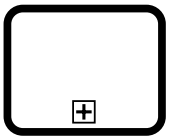
User Task
User Task is used to represent a task that a human should manually perform, either by the usage of a software application or by filling out a form. (Visual Paradigm, n.d.)



Service Task
Service Tasks utilize some sort of Web service or automated application to complete the task.
(Visual Paradigm, n.d.)



Script Task
An instruction-based task that contains a script that can be interpreted and executed by a BPMN engine, the task is considered complete when the execution of the script is complete.
(Visual Paradigm, n.d.)



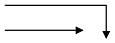
Call Activity
References a link to an already defined process, hence allowing for the re-use of processes that might take place in other parts of the organization. (Visual Paradigm, n.d.)

3.1.3 Gateways

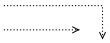
Gateway
Functions as a decisive step in the process flow, it decides how the flow should continue based on if certain criteria or conditions are met, if the conditions are not satisfied it can lead the flow in another direction.
(Lucidchart, 2022)

3.2 Connecting Objects

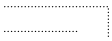
The connection objects are different types of lines that clarify the direction of the workflow, and or the connection/relation between different flow objects and artifacts. (Lucidchart, 2022)



Sequence Flow
Represents the sequential order and direction of the workflow. (Wondershare EdrawMax, 2022)



Message Flow
Visually depicted as a dotted line, Message Flows indicate how information is shared across the workflow. (Wondershare EdrawMax, 2022)



Association
References the relationship or association between other objects and artifacts. (Wondershare EdrawMax, 2022)

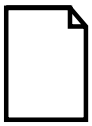
3.3 Swimlanes

Swimlanes are considered as one of the most essential elements in the definition of a BPMN workflow, swimlanes or sometimes referred to as a Pools, acts as a container that makes use of rows to visually organize a business process, these rows are called Lanes which define who will ultimately perform certain activities, while pools refer to a broader division like “marketing” or “IT”, see figure below for an illustration of a pool consisting of two lanes. (Gliffy, 2020; Lucidchart, 2022)

NOVIA
UNIVERSITY OF APPLIED SCIENCES

3.4 Artifacts

Artifacts is a collection name for annotations, data models, and groups within a BPMN workflow, and are mostly used to describe the dependencies of different functions within an organization. (Gliffy, 2020)



Data
Indicates that a certain activity either requires a specific data input or that some information has been produced as a result of an activity, depending on the direction of the message flow from or to the data element. (Lucidchart, 2022)

Data Storage
Depicted with the database icon, data storage elements represent the flow of access to stored data that can be associated with the business process. (Lucidchart, 2022)

Group
Groups are used as an organizational tool to visually group certain activities, hence they do not serve any purpose during execution inside a BPMN engine. (Wondershare EdrawMax, 2022)

Text Annotation

Text annotation is mainly just a comment that can be attached to any flow object when additional information might be required for the reader of the business process. (Gliffy, 2020)

NOVIA
UNIVERSITY OF APPLIED SCIENCES

4 Bpmn-Js

Bpmn-js is an open-source JavaScript library developed by BPMN.IO, it allows for viewing and editing BPMN 2.0 diagrams in the browser, thanks to the well-rounded design it allows for great extensibility and customizability. The three main exports of the Bpmn-js library are the modeler, viewer, and navigated-viewer, the modeler comes shipped ready with the required tools for mapping out business workflows, while the viewer and the navigated-viewer are tools specifically made for the visual inspection of business workflows, the navigated-viewer compared to its counterpart registers mouse inputs thus allowing for the free-roaming of the business workflow while the viewer is completely static. (BPMN.IO, 2020)

4.1 Dependencies and other libraries

The BPMN-Io project has split some of its core services into external libraries, which then can be individually imported into other projects as well. The Bpmn-js library relies upon both the Diagram-js and the Bpmn-moddle libraries, mostly by directly extending upon the services of these libraries. (BPMN.IO, 2020)

4.1.1 Diagram-Js

Provides the ability to interact with the graphical elements and the modeling tools either by drawing shapes and connections or such as the context-pad, tool palette, and key functionalities like undo/redo. Some of the essential services used in the Bpmn-js stem from the Diagram-js library, but most of them have respectively been extended. (BPMN-IO, 2021)

4.1.2 Bpmn-Moddle

Handles the reading and writing of BPMN 2.0 diagram files. It utilizes the BPMN 2.0 meta-model to validate inputs to correctly write schema-compliant XML documents. It also allows for the retrieval of BPMN-related information behind the diagram’s elements and connections. (BPMN-IO, 2021)

4.1.3 Didi

Didi is a lightweight dependency injection container for JavaScript, created by Nico Rehwaldt. Existing as a well-maintained fork of a library called node-di, Didi serves as the backbone of the Bpmn-js library, by allowing the use of complex dependencies mappings, the library has a well-rounded design, where each dependency can be extended upon. (Rehwaldt, 2022)

4.2 Essential services

Some of the essential services used by the Bpmn-js library will be covered in this section, all of which are important to familiarize oneself with to get the full potential of the library. (BPMN.IO, 2020)

4.2.1 Canvas

Manages the viewport of both the Bpmn-js modeler and viewer and provides a variety of useful APIs, not only for controlling the actual canvas such as focus and zoom and scroll, but also things like layers and markers. The canvas container has been given a layered design, allowing for the rendering of so of others. (BPMN.IO, 2020)

4.2.2 Overlays

The overlay service allows for dynamically attaching HTML elements to any of the flow objects on the canvas. To attach an overlay, one should use the add method of the overlay service, where the first parameter should be the id of the flow object to whom the overlay is going to be attached, the second being a configuration object consisting of the HTML element that is to be attached, and also a position property which indicates how the overlay should be positioned relevant to the flow object, for example of attaching an overlay see figure below. (BPMN-IO, 2020)

```
var overlayHtml = $('<div>Mixed up the labels?</div>');

overlayHtml.click(function(e) {
  alert('someone clicked me');
});

// attach the overlayHtml to a node
overlays.add('SCAN_OK', {
  position: {
    bottom: 0,
    right: 0
  },
  html: overlayHtml
});
```

Figure 1. Illustration of how an overlay is attached.

4.2.3 Element Registry

A registry service containing all elements and their corresponding data within the business process, the element registry provides useful APIs for retrieving elements or their respective graphics by id. To obtain a specific BPMN element one should make use of the “get” method and pass the id of the requested element, as shown in the figure below. (BPMN.IO, 2020)

```
var elementRegistry = bpmnJS.get('elementRegistry');

var sequenceFlowElement = elementRegistry.get('SequenceFlow_1'),
```

Figure 2. Fetching an element based on its id.

Also worth mentioning is the “getAll” method which should be used if one would want to acquire all the elements contained within a business process, the registry service also contains a filter function that can be used when wanting to acquire all the elements that meet a specified criterion. (BPMN.IO, 2020)

4.2.4 Eventbus

Designed with a fire and forget policy the eventbus functions as the main communication channel between different services/modules of both the Bpmn-js modeler and viewer. The eventbus is by design able to hook into the life cycle of the modeler or viewer, as well as handle the interaction of diagram from the end-user, thus allowing for more modularize functionalities which make it possible for new features to easily hook into pre-existing behavior of the library. (BPMN.IO, 2020)

A listener can be set up by the viewer or modeler's built-in support for directly contacting the eventbus or the injector can be used to obtain the eventbus service, as seen in the figure below. (BPMN-IO, 2020)

```
var eventBus = viewer.get('eventBus');

// you may hook into any of the following events
var events = [
  'element.hover',
  'element.out',
  'element.click',
  'element.dblclick',
  'element.mousedown',
  'element.mouseup'
];

events.forEach(function(event) {

  eventBus.on(event, function(e) {
    // e.element = the model element
    // e.gfx = the graphical element

    log(event, 'on', e.element.id);
  });
});
```

Figure 3. Illustration of how different life-cycle events can be hooked into.

The four main methods of the Eventbus are the following:

- On
- Once
- Fire
- Off

The “on” and “once” method are almost identically in the way how they function, both attach a listener which is set to listen to a specific event and captures the context of the fired event, the key difference between these methods is that the “once” method will remove its listener after the capturing of a single event, however, active listeners can also be removed manually by using the “off” method. (BPMN.IO, 2020)

When wanting to invoke active listeners the “fire” method should be used, said method takes the name of the event and a context object as its parameters, and the context object should contain information about the event that could potentially be extracted by the active listeners. (BPMN.IO, 2020)

4.2.5 BPMN Factory

As the name suggests the BPMN factory functions as a factory for creating shapes and connections based on the defined internal meta-model, in other words, it ensures that the creation of different elements follows the correct pattern, therefore it can be used to produce elements that do not yet exist in the actual XML file. (BPMN.IO, 2020)

4.2.6 Command stack

The command stack is responsible for processing the execution of commands, by transitioning them through different stages, this is achieved by the firing events on the previously mentioned eventbus, therefore allowing for other services to hook into the life cycle of said events. (BPMN-IO, 2021)

A command will first be validated by the “canExecute” method on the command stack before its execution, when a command is to be executed it will transition through these life-cycle events listed below. (BPMN-IO, 2021)

- preExecute
- preExecuted
- execute
- executed
- postExecute
- postExecuted
- revert
- reverted

In the illustration below one can see a simplified event life-cycle for the creation of a BPMN element, and how each of these life-cycle events is being fired respectively on the eventbus with services hooked into them.

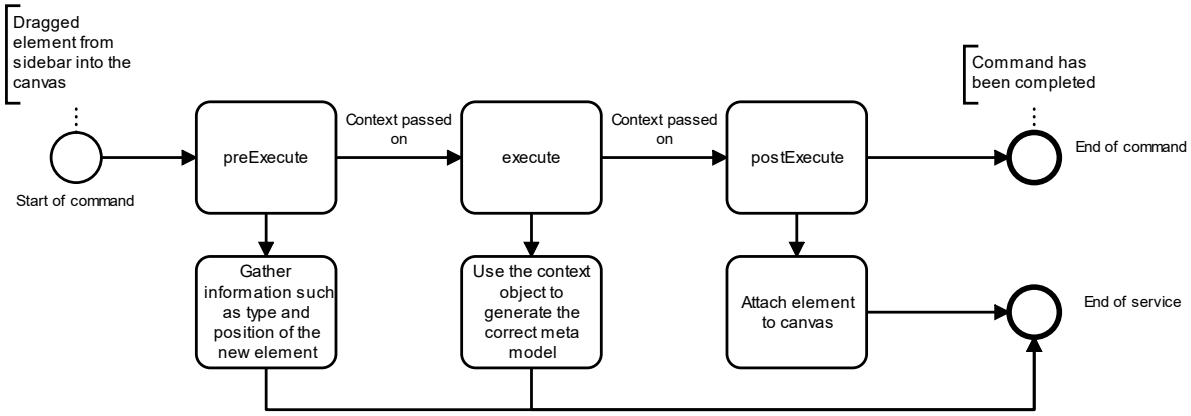


Figure 4. Simplified illustration of the element creation event life-cycle

4.2.7 Graphics Factory

Consists of multiple renderers the graphics factory is responsible for creating the graphical representations of shapes and connections, this is done through placing active listeners on the eventbus, when the desired event is captured the graphics factory is activated and renders depending on what is requested in the event. (BPMN.IO, 2020)

One example is when a diagram has been fully parsed in other words when the data has been fully loaded from a BPMN file, the graphics factory will initiate the rendering of all the elements within the element registry, the elements are passed over to the renderers. (BPMN.IO, 2020)

NOVIA
UNIVERSITY OF APPLIED SCIENCES

4.2.7.1 Base Renderer

The base renderer stems from the diagram-js library and functions as a generic boilerplate for renderers created with the diagram-js library, the base render implements a simple data model consisting of only shapes and connections, which can be seen in the figure below. (BPMN.IO, 2020)

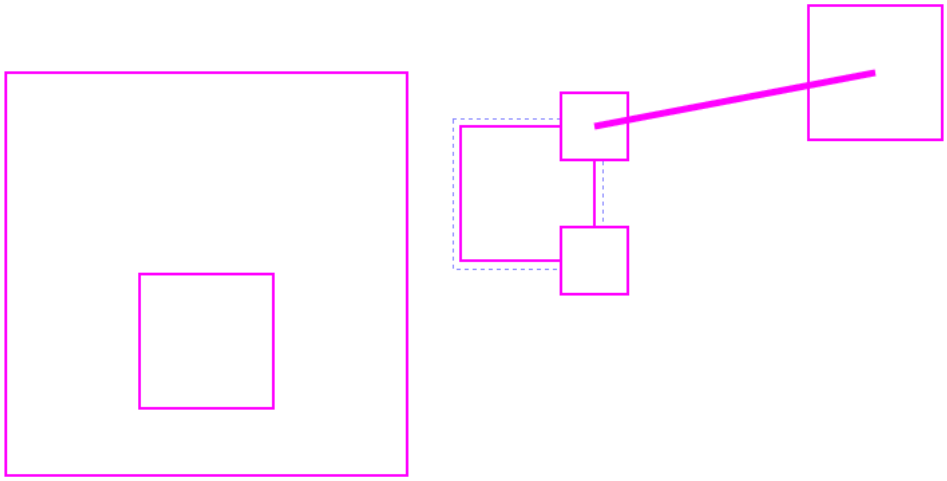


Figure 5. Illustration from BPMN-IO

4.2.7.2 Default Renderer

The default renderer is also a service of the diagram-js library, it inherits the base renderer and continues to add colors to connections, and gives the actual look and feel of the BPMN-IO projects. As seen in the figure below the default renderer has the lowest level of priority, therefore it only renders in case all the other renderers fail to do so. (BPMN-IO, 2021)

```
// apply default renderer with lowest possible priority
// so that it only kicks in if noone else could render
var DEFAULT_RENDER_PRIORITY = 1;
```

Figure 6. The render priority of the default renderer

4.2.7.3 Bpmn Renderer

The bpmn renderer is responsible for the rendering of all BPMN-related content, thus making it unique to the Bpmn-js library, in their other projects such as the DMN-js library this renderer is replaced by the DMN renderer. (BPMN-IO, 2021)

This renderer consists of methods for both drawing generic shapes and specific types of elements, for example, the figure below is the generic method for drawing rectangles used for the rendering of different types of tasks. (BPMN-IO, 2022)

```
function drawRect(parentGfx, width, height, r, offset, attrs) {

  if (isObject(offset)) {
    attrs = offset;
    offset = 0;
  }

  offset = offset || 0;

  attrs = computeStyle(attrs, {
    stroke: 'black',
    strokeWidth: 2,
    fill: 'white'
  });

  var rect = svgCreate('rect');
  svgAttr(rect, {
    x: offset,
    y: offset,
    width: width - offset * 2,
    height: height - offset * 2,
    rx: r,
    ry: r
  });
  svgAttr(rect, attrs);

  svgAppend(parentGfx, rect);

  return rect;
}
```

Figure 7. The method used when rendering different types of tasks

NOVIA
UNIVERSITY OF APPLIED SCIENCES

4.2.7.4 Custom Renderer

To either render custom elements or render default BPMN elements differently a custom renderer service must be created, this can be archived by extending the base renderer, and through extension, it will also be ensured that the custom renderer will be called whenever a connection or shape is to be rendered, it is important to note that the custom renderer will need a higher priority level than the base renderer since the custom renderer should always be called before the base renderer. (BPMN-IO, 2021)

```
const HIGH_PRIORITY = 1500;

export default class CustomRenderer extends BaseRenderer {
  constructor(eventBus) {
    super(eventBus, HIGH_PRIORITY);

    ...
  }
  ...
}
```

Figure 8. The higher prioritized custom renderer extends the base renderer

A set of rules is required for the custom render to be able to dictate which renderer is going to be rendering the shape or connection, this is done through the "canRender" method which will validate the shape or connection in the parameter of the method if the method returns true then either the renderer's "drawShape" or "drawConnection" method will be initiated depending on the type of the element, however, if the method returns false it will simply be passed down to the renderer with the next highest priority which in this case is the base renderer. (BPMN-IO, 2021)

```
canRender(element) {

  // only render tasks and events (ignore labels)
  return isAny(element, [ 'bpmn:Task', 'bpmn:Event' ]) && !element.labelTarget;
}
```

Figure 9. The method used to define if the renderer should render the element

The “drawShape” and “drawConnection” are methods of the renderers which contain the logic for how specific elements should be rendered, if one only wants a slight change in the appearance of an element it is wise to allow for the BPMN renderer to first render it normally and then alter the outcome as shown in the figure below. (BPMN-IO, 2021)

```
drawShape(parentNode, element) {
    const shape = this.bpmnRenderer.drawShape(parentNode, element);

    if (is(element, 'bpmn:Task')) {
      const rect = drawRect(parentNode, 100, 80, TASK_BORDER_RADIUS, '#52B415');

      prependTo(rect, parentNode);

      svgRemove(shape);

      return shape;
    }

    const rect = drawRect(parentNode, 30, 20, TASK_BORDER_RADIUS, '#cc0000');

    svgAttr(rect, {
      transform: 'translate(-20, -10)'
    });

    return shape;
  }
```

Figure 10. The usage of the BPMN renderer to later alter the element.

As shown in the figure above, the task is first rendered by the BPMN renderer and then replaced with a custom rectangle generated from the “drawRect” function, a big advantage of this method is that the labels and markers are left to the BPMN render handle. (BPMN-IO, 2021)

Finally, the “getShapePath” method handles the logic of how connection objects are attaching to elements, the method can be altered to give fewer attaching points or alter the radius of the attachment points from the center of the element, see figure below. (BPMN-IO, 2021)

```
getShapePath(shape) {
  if (is(shape, 'bpmn:Task')) {
    return getRoundRectPath(shape, TASK_BORDER_RADIUS);
  }

  return this.bpmnRenderer.getShapePath(shape);
}
```

Figure 11. The logic of how connection objects are attached to the element

4.3 Extension of the Bpmn-js library

The Bpmn-js library is built using the JavaScript object prototype structure, therefore it is possible for another JavaScript object to inherit all the methods and properties of a pre-existing prototype object, thus allowing for the behavior of any service of the Bpmn-js library to be modified or extended, to extend a service a new service should be created and passed to the constructor of the service which is to be extended. If an internal service is to be completely overwritten, the new service should be passed through the dependency injection system carrying the name of the service which is to be overwritten.

4.4 Bpmn File format

The file format used for BPMN workflows are notated as ".bpmn" and uses XML encoding for its file structure with the root element being the definitions element which contain information about the process as a whole, the actual BPMN elements are divided into metadata and graphical data, the graphical data only contain information such as position, width and height for elements and can be distinguished by the "bpmndi" prefix, while the metadata are stored either under a process or collaboration element depending on if the diagram contain any swimlanes, this is because the collaboration element functions as a container for the swimlanes to be stored under, therefore it is important to note that if a swimlane is added to the diagram the other services will consider the collaboration element to be the root element, hence additional information stored in the process element will not be carried over to the collaboration element, hence it is important to store information that is not tied to the actual process element in the definitions element instead, see figure 10 and 11 below for an illustration of the file format. (Flowable, 2022)

NOVIA
UNIVERSITY OF APPLIED SCIENCES

5 Design studio

Design Studio is a developed web application functions as a dedicated BPMN platform for both business process modeling and collectively storing them, Design Studio provides Wartsila with highly customized BPMN tools required by the organization for the creation of properly structured and well-documented processes, all with a clean and user-friendly design in mind.

Design Studio is built as a single page application (SPA) with Angular and fitted with a .NET backend, the application went through many overhauls where both the frontend and backend were changed, during initial development the backend was created with Node.js but was rather quickly changed to .NET to avoid relying on many external dependencies because of the unpredictable nature of the NPM package manager, along with the .NET backend the frontend was switched also switched to ASP.NET which was used during the pre-stage development and mostly used as a proof of concept on different features when eventually the project was greenlighted and the development proceeded ASP.NET was replaced by Angular.

5.1 Angular

While ASP.NET did not pose any issues, Angular was much more fitting for the project as one could make use of the component-based structure of Angular to create reusable components, therefore enabling the development of much more complex features, such as wrapping the Bpmn-js viewer inside a component and reusing it whenever a diagram is wished to be viewed on a web page.

5.1.1 Bridging Angular services to Bpmn-js

A dependency bridge was created as it was not possible to utilize any of the Angular services from within the Bpmn-js modeler, this was achieved by bridging the Angular and Bpmn-js dependency injection systems, the bridge takes an array of Angular services and wraps them as additional modules for Bpmn-js. Among the bridged services are the Angular router and HTTP client, the router has much better overall performance due to its utilizing lazy loading, and the HTTP client allows for the use of HTTP interceptors.

5.1.2 HTTP interceptors

Taking use of Angular's HTTP-interceptors simplified API calls, by allowing one to intercept any frontend API call and check if an error code is returned, if there is an error it can be generically handled, for example, if an unauthenticated user is trying to access an endpoint which requires the user to be authenticated the API call will return with an error of status 403, and the user would be presented with a blank screen, therefore a catch is needed to ensure the user is not left confused.

When using HTTP interceptors there is no need to write catches on every API request but instead use an interceptor that can check what error the request returns with and respectively handle the error, therefore if any API call returns with the error 403, the user can navigate to the sign-in page.

5.2 Component structure

As the Bpmn-js viewer is used on different webpages of the application, depending on the use case different services can be required when the viewer is initiated, therefore both the Bpmn-js viewer and modeler have been placed in components respectively, allowing them to be injected into other components and controlled by the parent component, therefore the configurations are stored in the parent component.

5.2.1 Hierarchy

The hierarchy is a navigable folder tree, consisting of two roots a private and public root, the public root is used for official business processes, while the private root consists of nonofficial processes such as used for training new users or personal process drafts.

The hierarchy component can also be embedded into other components when there is a need to obtain a specific folder or business process from storage, for example when a user has begun to model a business process but has not yet saved it to any specific location, the hierarchy components were imbedded in a popup to allow the end-users to choose a location for where the process should be stored.

5.2.2 Designer

The designer component is responsible for all of the process modeling, it's made up of three child components, therefore it functions as a component container for the Bpmn-js modeler, properties panel, and the toolbar, as the designer is the parent it also serves as a communication channel between the different child components.

The sequence of component initialization is very important because both the toolbar and properties panel respectively rely on communication to or from the Bpmn-js modeler, for this communication to be possible, each component needs to have a reference to the Bpmn-js modeler, and this causes an issue because the parent component is always the first to be imitated, and on its initiation, it will pass a reference down to the child components, but at this stage, the Bpmn-js modeler is not yet initiated and can't be passed onwards until it has been.

Achieving the communication was possible by making use of the Angular component life cycle, therefore the designer can make use of the Angular's "AfterContentInit" hook, which is a callback function that is called after the contents of a component have all been initiated, therefore this callback was used to pass down the Bpmn-js modeler reference trough to the other components.

5.2.3 Toolbar

The designer's toolbar includes a tab system where each tab represents a business process that is currently open in the designer component, thus allowing for an end-user to have multiple business processes open at once and switch between them by clicking on the tabs.

The toolbar functions by using the hierarchy component to allow end-users to select a business process, once a selection has been referred to that business process will be stored, and when switching between different processes the reference is then passed to the Bpmn-js modeler to open that process instead.

5.2.4 Inspector

The inspector is a parent container for the Bpmn-js viewer and a metadata panel. It functions as a component for process monitoring. A selected business process can be visually displayed in the Bpmn-js viewer, and its respective data can be used and viewed in the metadata panel.

The metadata panel visualizes the metadata on both process level and activity level, by clicking on an element of interest it updates the data shown in the metadata panel, thus allowing the end-users to easily filter for specific information.

Also implemented into the inspector is the ability for end-users to quickly navigate to business processes linked in call-activities, this was accomplished by displaying a popup of the linked process whenever the end-user hovers their cursor over a call activity, and by double-clicking the call-activity the end-user is navigated to the inspection of said process, thus creating a smooth user experience for mapping processes together.

Inspecting different versions of a process is also available in the inspector component, not only can the different versions of a business process be displayed but they can also be compared against each other, therefore it’s possible to see how a specific process has been changed over its lifespan, and notably if there are any performance differences, thus allowing for process optimization to take place.

6 Implemented features

During the development, multiple features were requested, with a handful of the requests coming from the end-users themselves, therefore each of the requests was thoughtfully considered before any actual development was done to ensure that the BPMN standard was followed, in this section each of developed features will be covered in more depth.

NOVIA
UNIVERSITY OF APPLIED SCIENCES

6.1 Properties panel

The prebuilt properties panel developed by Camunda gives the end-users the availability to edit more technical process properties, some of the properties are only essential when doing process modeling on a deeper level, and for less BPMN experienced users the panel only appears cluttered and confusing.

The drawbacks of the prebuilt properties panel lay in that the panel is being injected into the DOM by the usage of external libraries such as the “domify” method from the min-dom library, which takes an HTML string and then proceeds to create the HTML elements. The “domify” method is also not capable of parsing elements from external libraries such as Angular Material or Bootstrap, therefore making it rather complicated to fit into the project structure visually.

As the unnecessary properties of the panel only confuse the average end-user, it was decided that the panel should be recreated as an Angular component and stripped of anything unnecessary, thus allowing for the implementation of more complicated features to be embedded into the properties panel, such as a markdown editor and the simplifying process linking with call activities.

6.1.1 Markdown editor

Replacing the existing element documentation text-area input with a markdown editor allows the end-users to write documentation in more detail, as markdown text supports the usage of headers, bullet lists, hyperlinks, etc.

The properties filled into the properties panel are attached to the selected BPMN element and can therefore be extracted when inspecting a business process, which also allows the end-users to link processes together with the usage of hyperlinks, for example when wanting to add additional information from an external source it can be attached as a hyperlink to the element’s documentation.

NOVIA
UNIVERSITY OF APPLIED SCIENCES

6.1.2 Call activity

The prebuilt properties panel had a specific input field for handling the call activity elements, where the end-users had to fill in the id of the targeted business process, which was reported by the end-users to be quite a tedious task since one had to either remember or look up the business process, this was solved by having the input fields replaced with a button instead, when clicked the previously mentioned hierarchy component is presented inside a popup which allows the end-users to navigate the hierarchy and select the process that is wished to be linked, the information that would normally have been manually put into the input field can now be parsed from the end-users selection and automatically filled in.

6.2 Collaboration feature

The collaboration feature intends to allow users to simultaneously edit a business process, this was accomplished by inserting a collective of services into the Bpmn-js modeler, most notably the Collaboration, Creator, and Executer services.

The Collaboration service acts as the main socket controller by either sending or receiving commands between other users, where each client is connected to the backend of the application and can therefore be placed in a lobby with other users who are also currently editing the same process, thus allowing for commands to be sent within the lobby where the backend can recognize and validate to which other connected users the command should be passed on to.

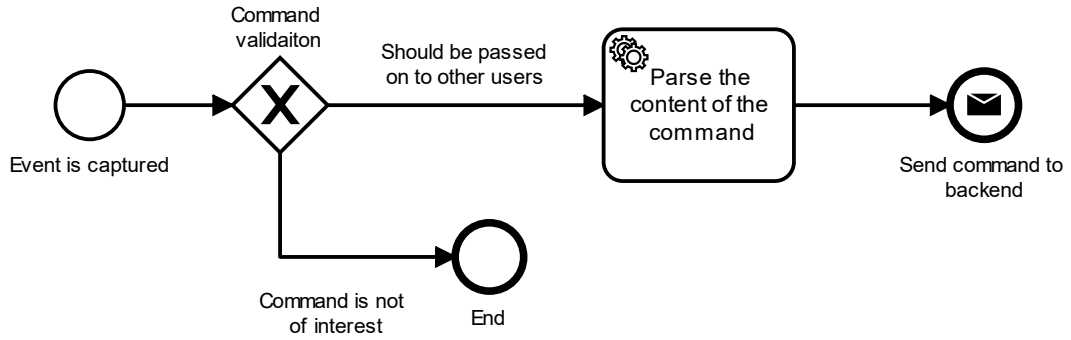


Figure 12. Illustration of a client initiating a command

As seen in the figure above the command capturing is done through utilizing the eventbus capabilities of listening on internal events, because any action done by a user will run through the command stack service, as mentioned previously the command stack always fires the command name followed by the state it is currently in, therefore allowing for the capturing of commands at a pre-execute state.

The pre-execute state holds the required information about a command needed to be able to execute the command, which means that the captured event's information can be used to replicate the same result on other clients, it is also important to note that commands can sometimes be chained together, thus making it very important to filter through and only capture events that start the chain.

However, using the pre-execute method is problematic, the issue is with the JavaScript language itself, as it always stores values by reference unless told otherwise, which means that the given command information will mutate as the command progresses through the different stages, this is unwanted behavior because it is not possible to replicate the command on another client after it has already been completely executed, therefore it is desired to have the initial values of the command to be able to follow the same set of exactions, to overcome this issue it's necessary to break the reference of the command object, this was achieved by passing the command to the Creator service which is responsible for configuring the command to be both passable and executable on the other client.

The Collaboration service also receives commands from the backend, which are then passed from the Collaboration service to the Executor service for execution which differentiates commands by name, the Executor will then call different functions inside the Bpmn-js library depending on the command to allow for the Bpmn-js library to do the heavy lifting, thus also allowing any necessary command chain to be started, for example, the move command used to change the location of an element on the diagram will also cause any arrow connected to the element to move.

It's important to note that the received command will be executed and therefore it will eventually be picked up by the eventbus listener, thus the same command will be sent over to the client it originated from, and by doing so creating an infinite loop of the same command, to prevent the infinite loop a creator property was attached to the command, thus marking the command with client's socket id, which can then be checked once the command has been captured, if the creator is not this current client, then the command will be caught in the command validation, see figure below for an illustration of an infinite command loop.

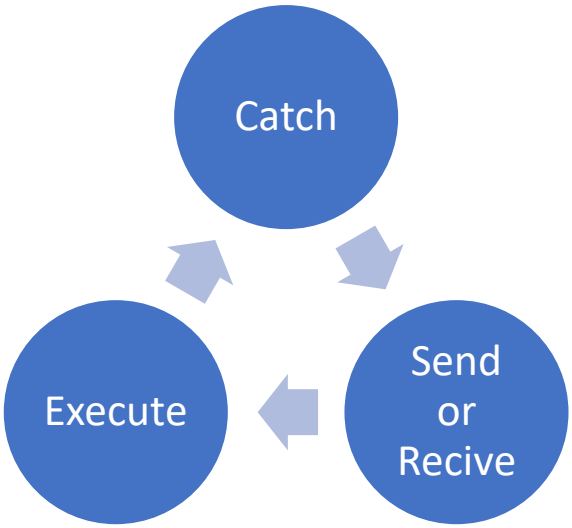


Figure 13. demonstration of an infinite command loop

In the event of unsuccessfully executing a received command, a rollback request is sent to the client the command originated from and requests the whole XML, which can then be opened to re-sync the users, thus assuring that none of the clients will go out of sync with each other.

6.3 Group element

A new visual element in the shape of a vertical dotted line was requested, the new line element would enhance the visual representation of milestones in a swimlane, but since such an element does not exist within the BPMN standard it could not simply be implemented as a new element; therefore it was decided to modify the group element instead.

The group element was chosen because it serves no other purpose than to visually group elements together, therefore it is only required to allow the group to change visually, this was done by making use of extension elements, which function as external data fields for elements.

An extension element containing a Boolean property was added to the group element, if said Boolean is set to true, then the renderer module knows to render the group as a dotted line instead, which allows users to toggle how the element is represented, and thanks to the group elements being pre-fitted with scaling that allow the group element to be resized on both the x and y-axis, the height of the line can then simply follow the height of the group element and then use a pre-defined width, thus allowing it to only scale on the y-axis.

6.4 Predefined lane names

During the initial user testing phase of Design Studio, it became clear that the pool lane naming could pose a potential problem, as users wanted to reference a specific role or unit in the organization but would occasionally misspell the names or reference them by plural, for example,” programmer” and ”programmers”, the problem occurs when end-users are unaware of the correct naming practices, this issue could not simply be solved with a static naming list as it wouldn’t update when the business structure changes and thus becoming a maintainability issue.

It became clear that the best way to solve the lane naming issue was to supply the end-user with a predefined list of names for the user to choose from, to allow the list to be easily updated and maintained, a dashboard view was created which contains all the roles and units in the organization for the department owners or other admins to maintain, the list could then be fetched over API from within the Bpmn-js modeler.

A new roles-and-responsibilities service was created to detect when an end-user is writing the name of a lane or collaboration element, when the name is being edited it triggers a filtered dropdown containing the names of roles and units from the dashboard to be shown, and the filtering is based on what the user has typed into the name input field.

6.5 Process variance

As business processes are descriptive diagrams explaining how an organization is operating and which departments are involved, sometimes the way of working within a department can differ depending on factors like the location of the department, for example, a pizza restaurant chain who have locations all around the world could have a business process called “Making pizza”, the way of working would be very similar between Helsinki and New York but with a slight change to some of the aspects of the process, for example in Helsinki they bake the dough themselves while in New York they buy it of a bakery, and that’s where process variance is needed.

Instead of having two different business processes that contain almost identical workflows, they can be stored as different variances of the same business process, accomplishing was done by adding a process variance meta-model to the database, which contains an id reference to the business process it belongs to and can therefore be fetched from the database alongside the main process.

6.6 Versioning

Version control is an important aspect of the business process lifecycle because an unattended business process may over some time no longer reflect the current way of working, and simply editing and updating a business process it would not allow for Wärtsilä to reflect upon how the organization used to operate, therefore it was decided that the application needs to be able to store previous versions as well as the new version of a business process.

Versioning was accomplished by first no longer supporting the possibility to edit a process after its creation, but this in return created a complication for end-users when creating the first version of a process, since they should be able to work on the first version of the process without having to rely on versioning, for example if an end-user creates a business process and then accidentally leaves a typo or any other kind of mistake they would now need to fix those mistakes with a new version, hence the first version containing mistakes is technically not the first version of the business process but instead just a draft that the end-user wants to have

deleted, this was solved by adding a Boolean state to the business process allowing the end-users to mark their business processes as “official” which indicates that the draft of the business processes is completed and that the first version of the business process should be created, thus allowing for future updates to go through version control.

When end-users want to create a new version of a business process it’s required that the end-user specify which variance of the process is to be updated, thereafter the end-user can select from which version the draft is to be generated since older versions can sometimes be more relevant to use than the current version of the business process.

When the draft is considered complete the end-user can mark the draft for review, thus creating a review request for the process owner who then makes a decision based on the changes if it should be rejected or if the changes are considered to be only a minor revision or if it should be a new version, each version consisting of a version tag which is a numeric value to identify the version, minor revisions only increment the last two numbers of the version tag while a new version increments the first number and resets the others, for example, if the current version of a process is 1.2.6, a minor revision would bring the version tag to 1.2.7 while a new version would bring it to 2.0.0, see figure below for illustration.

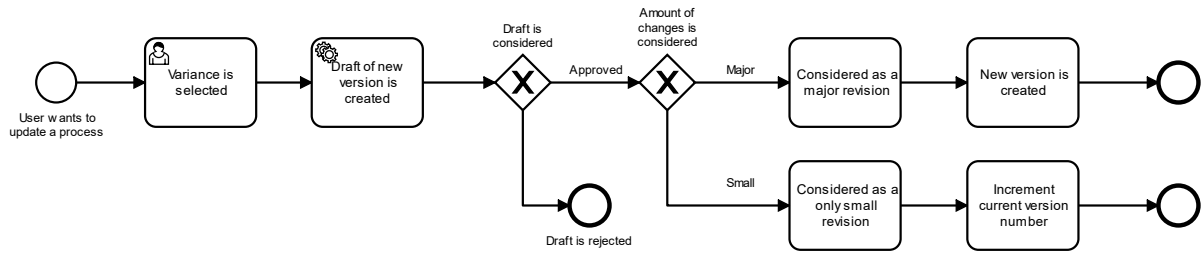


Figure 14. Illustration of the version control.

NOVIA
UNIVERSITY OF APPLIED SCIENCES

6.7 Process comparison

A process comparer was created to give decision-making assistance to the process owner during a review of a version request, the new version attached to the request will be compared against the current version of the process.

The comparer detects changes through a library called “Bpmn-js-differ” developed by BPMN-IO, which is a library dedicated to allocating differences between two business processes, to help the owner the comparer can also advise on the severity of the changes detected by the said library. The changes are considered a major or small revision based on the total amount of changes and the types of changes, small revisions often only contain changes in the BPMN elements such as text or descriptions, while major revisions often contain changes such as the removal or addition of new BPMN elements.

The comparer contains two Bpmn-js viewers that can respectively show the draft and the current version at the same time while highlighting changes that have been detected, these highlights come in the form of overlays and indicates based on colors what type of changes has been done to each element, green represents a new element while red indicates that an element has been deleted, and lastly orange to show that some values of that element have changed, thus aiding the owner to allocate the differences quicker.

NOVIA
UNIVERSITY OF APPLIED SCIENCES

7 Discussion

To have been able to take part in the development has been such a great learning experience, it's been equally motivating as it has been challenging but most of all it has been a very positive experience even though the scope of the thesis has felt quite large at times.

The Design-Studio has from time to time proven its value and capabilities, therefore I am very happy with the result of the application. The most interesting part of the thesis has been to try and overcome and deliver features that have been requested by the end-users, as some of the features are quite complex and therefore require a lot of problem-solving.

7.1 Future

Even though Design-Studio is mostly complete there are still possibilities for improvements, such as trying to bridge the project closer to other applications within Wärtsilä to try and gain as much benefit from the application.

7.2 Conclusion

I have had the opportunity to partake in both the development and the project planning as my previous manager left Wärtsilä in the middle of the development, which allowed me to fill his shoes for this project. Thus, I have been able to speak directly to the end-users which has been the most rewarding part. Trough out it has been a great learning experience, as it has been an eye-opening experience as I have been able to see the full cycle of software development, from the project planning to implementation and lastly getting the feedback from the end-users. Lastly, I would like to thank my co-workers and managers for helping me along the way and giving me this experience, thanks to the team I look forward to a long career in software development.

NOVIA
UNIVERSITY OF APPLIED SCIENCES

8 References

Angular. (2022). *AfterContentInit*. From Angular.io: https://angular.io/api/core/AfterContentInit

BPMN.IO. (2020). *walkthrough*. From bpmn.io: https://bpmn.io/toolkit/bpmn-js/walkthrough/

BPMN-IO. (2020, May 28). *interaction*. From github: https://github.com/bpmn-io/bpmn-js-examples/tree/master/interaction

BPMN-IO. (2020, May 28). *overlays*. From github: https://github.com/bpmn-io/bpmn-js-examples/tree/master/overlays

BPMN-IO. (2021, March 29). *bpmn-js-example-custom-rendering*. From github: https://github.com/bpmn-io/bpmn-js-example-custom-rendering

BPMN-IO. (2021, April 19). *bpmn-moddle*. From github: https://github.com/bpmn-io/bpmn-moddle/blob/master/README.md

BPMN-IO. (2021, January 22). *CommandStack*. From github: https://github.com/bpmn-io/diagram-js/blob/master/lib/command/CommandStack.js

BPMN-IO. (2021, November 2). *diagram-js*. From github: https://github.com/bpmn-io/diagram-js

BPMN-IO. (2022, February 9). *BpmnRenderer*. From github: https://github.com/bpmn-io/bpmn-js/blob/master/lib/draw/BpmnRenderer.js

Bpmn-Io. (n.d.). *Canvas.js*. From github: https://github.com/bpmn-io/diagram-js/blob/master/lib/core/Canvas.js

Flowable. (2022). *BPMN 2.0 Introduction*. From flowable: https://www.flowable.com/open-source/docs/bpmn/ch07a-BPMN-Introduction

Gliffy. (2020, December 11). *guide-to-bpmn-symbols*. From gliffy: https://www.gliffy.com/blog/guide-to-bpmn-symbols

Lucidchart. (2022). *bpmn*. From lucidchart: https://www.lucidchart.com/pages/bpmn

Lucidchart. (2022). *bpmn-symbols-explained*. From Lucidchart: https://www.lucidchart.com/pages/bpmn-symbols-explained

Lynch, A. (2022, February 18). *what-is-bpmn*. From edrawsoft: https://www.edrawsoft.com/what-is-bpmn.html#history

PubNub. (n.d.). *what-is-signalr*. From pubnub: https://www.pubnub.com/learn/glossary/what-is-signalr/

Rehwaldt, N. (2022, March 4). *didi*. From github: https://github.com/nikku/didi

Visual Paradigm. (n.d.). *bpmn-activity-types-explained*. From visual-paradigm: https://www.visual-paradigm.com/guide/bpmn/bpmn-activity-types-explained/#bpmn-tasks

Visual Paradigm. (n.d.). *what-is-bpmn*. From visual-paradigm: https://www.visual-paradigm.com/guide/bpmn/what-is-bpmn/

Wärtsilä. (n.d.). *www.wartsila.com*. From history: https://www.wartsila.com/about/history

Wondershare EdrawMax. (2022). *bpmn-symbols*. From edrawmax: https://www.edrawmax.com/article/bpmn-symbols.html