TU WIEN Informatics

How to Model and Transform Executable BPMN Process Models

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Software & Information Engineering

by

Dragana Sunaric
Registration Number 11814569

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Inf. Dr.-Ing. Jürgen Dorn

Vienna, 1st January, 2022

Dragana Sunaric	Jürgen Dorn

Declaration of Authorship

Dragana Sunaric

I hereby declare that I have written this Doctoral Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

Vienna, 1st January, 2022

Dragana Sunaric

Acknowledgements

write Acknowledgements

Abstract

write Abstract

Contents

CHAPTER 1

Introduction

Business Process Model and Notation (BPMN) is an widely used and established diagramming language with the purpose of visualizing business processes in organizations. Process models are usually created by process analysts together with domain experts. BPMN-models created that way contain incomplete and implicit information for the specific business domain and are therefore called conceptual BPMN model. By involving IT-experts these conceptual BPMN models can be made executable and automated using a Workflow Managment System (BPMN) Workflow Management System (WfMS). [11]

Today WfMSs like Camunda BPM and Alfresco Process Services (BPM) are widely used by big organizations [3] [1]. Real-life processes being executed in WfMSs can be complex and have to be changed and reevaluated together with the software embedded in these processes as requirements and guidelines for the specific domain change.

One advantage of directly automating processes defined as BPMN is its readability for IT-experts and domain-experts. As a consequence both parties have a common understanding of the process that is executed and have basis for discussion when changes have to be made. In order for the cooperation between IT and business to work smoothly, executable BPMN model should be as simple to understand as possible and be in line with known standards and best practices.

Applying guidelines for good executable BPMN models does not only increase readability but can also have direct impact on process costs. Due to the pricing scheme of some enterprise WfMSs, which take into account the total number of nodes passed by process instances, eliminating non necessary handovers and therefore reducing the number of nodes in the BPMN model, can have an impact on the required license.

1.1 Goal of This Thesis

The goal of this thesis is to give practical guidance on how to apply best practices and methodologies to create good executable BPMN models and to improve existing models.

This will be achieved by stating the current research on creating executable BPMN models. After that this work will take a look at methods for transforming and improving existing executable BPMN models and how to use them to create models in line with the stated best practices. In order to compare two process models, this work will also provide methods for measuring metrics like time and cost of processes defined as BPMN models. Finally this thesis will provide detailed information about how to use these methods in practice with a case study.

1.2 Structure of This Thesis

This Thesis starts with an state-of-the-art section consisting of two chapters. The first one is about the nature of executable BPMN models and how to make an existing conceptual BPMN model executable by an WfMS. The second chapter provides methods for analyzing and improving processes using Six Sigma approaches and quantifiable measures.

Along with this thesis a software was developed which suggest changes to an executable BPMN model given as an Extensible Markup Language XML (XML)-file based on the best practices found in the first two chapters. The documentation and implementation details on this software can be found in chapter 4.

Not every aspect of transformation given in the first two chapters is suitable to be fully automated and requires knowledge on the specific domain of the process. Therefore, the last chapter provides a case-study to an existing process model applying the principles from the first two chapters and giving a guide on how to use those principles in practice.

CHAPTER 2

Automating Business Processes

The following chapter will provide an overview of the current research on executable BPMN models. First it will discuss the motivation behind process automation and using WfMSs and the benefit it might bring to an organization. Then it will state the differences between an conceptual BPMN model, that cannot be deployed on an WfMS, and an executable BPMN model. Finally this chapter will list the steps necessary to turn an conceptual BPMN model executable.

2.1 Why automate Processes?

Introducing an WfMS in an organization is often done in order to open the possibilities to automating parts of a process or even the whole process. While process automation is an perused goal, there are some quantifiable advantages to using a WfMS besides the possibility to automate certain process steps.

- **Shorten process lifetimes**: Managing processes manually requires handling tasks such as starting sub-processes or handling handovers from one entity to another and can lead to unnecessary waiting periods between two tasks. By implementing an WfMS resource allocation and parallelization can be automated where possible to assure optimal use of process resources. [13]
- **Reduction in process cost**: By reducing process lifetimes and increase productivity due to better handling resources can reduce process costs [13] but due to the high price for (BPMN) Workflow Management Systems the overall cost does not have to decrease necessarily [14].
- **Workload reduction**: As stated earlier, managing processes needs to be done either by an (BPMN) Workflow Management System or manually which creates

additional workload for an organization. Workloads for employees executing the processes are also kept steady due to dynamic resource assignment. [11][13]

- **Enforce rules**: Defining the processes that is then directly executed and controlled by an (BPMN) Workflow Management System enables the organization to enforce the execution of the process at it is designed. Following Rules and Protocols can partially be automated and enforcing guidelines and laws in an organization becomes easier. [11]
- **Create Transparency**: Using an (BPMN) Workflow Management System provides insights to the actually processes that are executed in the organization. It makes it easier to determine the performance of the processes by providing historical information of completed process instances and provides insight of the current status of processes that are still in progress. [13]

2.2 Executable vs Conceptual Process Models

Process Models are inherently Business-oriented as their purpose is initially to define and visualize the processes happening in an organization. Business-oriented or conceptual BPMN model are meant to be read by domain experts and contain usually implicit information known to these domain experts [11]. They are usually incomplete, meaning that not every possible outcome of a process is modeled. In fact, usually only the best-case scenario, also known as the 'happy-path', of a process is modeled in an conceptual BPMN model [12].

Because of this, business-oriented models cannot be executed in an (BPMN) Workflow Management System as they are but have to be turned into an executable BPMN model. executable BPMN model are meant for IT-experts and should be a technical representation of the business process while still begin understandable by domain experts. They should leave no room for interpretation as they have to contain all the information necessary for the process to be executed using an (BPMN) Workflow Management System. Besides the visual information, executable BPMN model also need to contain execution properties like interface definitions and Variables that are used by the Process called Process Variables.[11]

An algorithm on how to efficiently turn an conceptual BPMN model executable as stated in the book *Fundamentals of Business Process Management* [11] is described in the next section.

2.3 Making Process Models executable

As stated earlier, BPMN models can not directly be executed by a Business Process Management System Business Process Management System (BPMS) but have to be converted from an conceptual BPMN model into an executable BPMN model.

There are different approaches how an executable BPMN model can be derived from a business-oriented conceptual BPMN model. In [11] performing such a transformation is broken down into 5 steps:

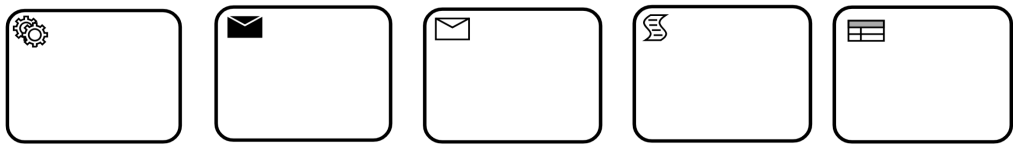
1. Identify the automation boundaries
2. Review manual tasks
3. Complete the process model
4. Bring the process model to an adequate granularity level
5. Specify execution properties

2.3.1 Identify the automation boundaries

The first step in turning an conceptual BPMN model in an executable BPMN model is to identify which steps can be automated using a (BPMN) Workflow Management System.

Tasks which can inherently be automated are called Automated tasks [11, p. 317]. Taking a look at the BPMN 2.0 standard an Automated task can be one of the following Task types:

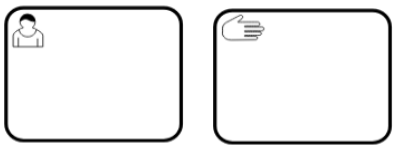
- **Service Task**: A Task that invokes a service. Can be a Webservice or a application code.
- **Send Task**: Used to send a message to an external participant (A participant that is not part of the process)
- **Receive Task**: Used to receive a message from an external participant
- **Script Task**: Executes a Script that can be interpreted by the (BPMN) Workflow Management System
- **Business Rule Task**: Executes a rule. (e.g. provides input for a business rule engine and gets the output of that calculation)



(a) Service Task (b) Send Task (c) Receive Task (d) Script Task (e) Business Rule Task

Figure 2.1: Automated Tasks according to the BPMN 2.0 standard [18]

Usually not every step of a Process can be fully automated. Processes can also have **Manual Tasks** and **User Tasks**. A **User Task** is a Task performed by a User with the aid of an (BPMN) Workflow Management System while a **Manual Task** does not use any help from a business process execution engine.



(a) User Task (b) Manual Task

Figure 2.2: Manual and User Tasks according to the BPMN 2.0 standard [18]

2.3.2 Review manual tasks

As mentioned earlier Manual Tasks are not automated and do also happen without the aid of a (BPMN) Workflow Management System. In this step we need to analyse if the identified manual tasks in our BPMN-model can be incorporated into the (BPMN) Workflow Management System. This can be done in two ways:

- **Automate the task**: Depending on the nature of the task that is currently performed manually, it might be possible to fully automate the task or to model the task as a Receive Task where the modelled task waits for a message that indicates the physical manual tasks completion.
- **Turn it into a User task**: If the task cannot be automated or the organization lacks the resources to automate the task yet, it might be possible to turn the Manual Task into a User Task. One possibility could be to dedicate a person in charge of the manual task to notify the (BPMN) Workflow Management System on the completion of the task via a worklist handler. [22]

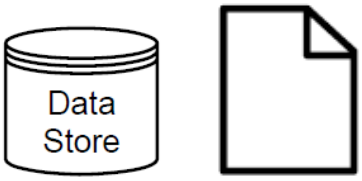
In the case that neither an Automated task nor an User Task is suitable for modelling the Manual Task one might also consider isolating the task and modelling the rest of the process. If this is also not possible due to the manual task being crucial for the expressiveness of the model it might be reconsidered if this process can or should be executed using a (BPMN) Workflow Management System[12, p. 228]

2.3.3 Complete the process model

Usually conceptual BPMN models are not complete and leave out certain informations that are seen as implicit knowledge or as not important by the person modelling the process but might be crucial if a complete picture of the process is needed for automation.

A common flaw in many conceptual models is ignoring errors and only implementing the 'happy path'. The 'happy-path' is the best-case scenario that can happen in the execution of a process. While it might be sufficient for a conceptual BPMN model, showing the process for a customer order, to not show what happens in case the product is out of stock or what happens if the payment does not work, a executable BPMN model has to take into account what happens in case an error occurs.

It is also necessary to model the input and output data of our tasks in this step using Data Stores and Data Objects.



(a) Data Store (b) Data Object

Figure 2.3: Data store and Data Objects according to the BPMN 2.0 standard [18]

2.3.4 Bring the process model to an adequate granularity level

The granularity of tasks modelled in an conceptual BPMN model does differ from the granularity needed in an executable BPMN model. The goal of process automation is not to automate as much as possible but to have a centralized (BPMN) Workflow Management System that does not only decide what task has to be done when but also who needs to be doing this task. [12]

Therefore consecutive tasks that are done by the same participant should be clustered together as one task to minimize handovers that have to be unnecessarily processed by the (BPMN) Workflow Management System. [11] However there are some exception to this rule:

- Tracking progress: In order to know how much the process has advanced it can be useful to split certain parts even if they are done by the same person.
- Handling exceptions: If for a set of tasks that it performed by the same participant different erros and exceptions can occur, it might be useful to keep these tasks separated.
- Managing Resources: Sometimes consecutive tasks are performed by participants that have the same role, but could to be done by two different participants in order to use resources better. In this case it is also to dis-aggregate the task accordingly.

2.3.5 Specify execution properties

The final step in turning an conceptual BPMN model executable is to specify the details of the implementation details of our BPMN model. While the changes performed up to this step had an impact of the graphical representation, the execution properties are not graphically embedded into BPMN but are encoded into the XML representation of the BPMN-model. [11] A schematic representation of the structure if BPMN is provided in 2.4. For the full specification of the BPMN-XML the XML-schema can be found on the OMG-Website[2].

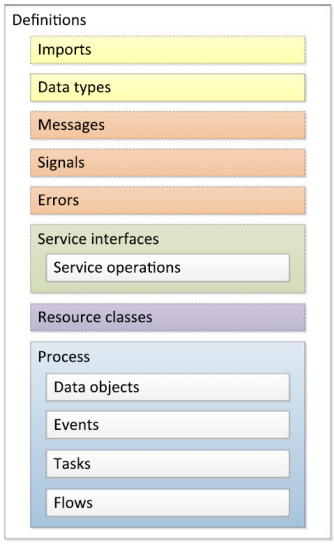


Figure 2.4: Structure of the BPMN format [11]

Process Variables
In order to use data in different elements of our process, we need process variables that can be read, created and modified during the processes execution. Every Process variable has a **data type** that can either be simple (strings, integers, doubles, booleans, dates, times, ...) or complex (composed of other types). A complex Type needs to be described as an XML Schema Definition XSD (XSD)-Schema File.

```
<xs:element name="person">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="name" type="xs:string"/>
                        <xs:element name="address" type="xs:string"/>
                        <xs:element name="city" type="xs:string"/>
                        <xs:element name="country" type="xs:string"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
```

Listing 2.1: The XML-Schema Definiton for a complex type 'person'

```
<person>
        <name>Dragana Sunaric</name>
        <address>Wiedner-Hauptstrasse 5</address>
        <city>1040 Wien</city>
        <country>Austria</country>
```

```
</person>
```

Listing 2.2: An instance of the complex type 'person'

The definition of common Errors, Messages and Escalations that are thrown or listened to by Events and Tasks are aslo part of the execution properties. Every Element has at least an **id** that identifies the given Element and a descriptive **name** of the element. [18]

Messages

```
<message id="Message_ID" name="Message_NAME"/>
```

Errors
Errors additionally have an **errorCode** that specifies the given Error. Events can listen for this specific error code and trigger when it is thrown.

```
<error id="Error_ID" name="Error_NAME" errorCode="Error_CODE"/>
```

Escalations
Similar to errors, escalations additionally have an **escalationCode** that specifies the given Error and can be listened to by events.

```
<escalation id="Esc_ID" name="Esc_NAME" escalationCode="Esc_CODE"/>
```

Input and Output Variables
As mentioned earlier, Process Variables are active during the whole Process life-cycle. Apart from this data than can be accessed globally, it is also possible to define input and output values for each Task or Event in our process-model. These values are only visible within the Task or Event and have to be defined as an XSD-Schema File (similarly to complex process variables). [11]

Service Tasks
In order for service tasks to call external application or web-services, the interaction with the given service has to be defined in the process model. Connected services need to provide an service interface that describes the available service-operations and their parameters as well as return values. Service-operations can be synchronous, meaning the process instance waits for the operation to finish and to return a value or error code, or asynchronous, meaning the process does not wait for a response and carries on with the process after calling the service. Based on the service interface definition, input and output variables have to be defined for the service-call. The (BPMN) Workflow

Management System does this by copying the above mentioned Input values of the Task into the service call and if necessary, copy the output values of the service call into the output values of the Service task. [11]

2.3.6 Apply Naming Conventions

It is recommended to apply naming conventions for BPMN 2.0 diagrams. While there are many suggestions for naming conventions, none is defined by the BPMN 2.0 standard [18]. The following is a suggestion for naming different BPMN elements based on [12], [**?**] [23] and [19].

Events
Names of Events should start with a business object (noun) followed by a verb. In order to indicate that something just happened, the verb should be in the past participle form. The noun can be specified by an adjective in front. Examples for good Event names:

- Order delivered
- Large Order delivered

Tasks
Tasks should be named starting with a verb and followed by a business object (noun). The object can be specified using an adjective. If needed, the task name can also answer how the task will be accomplished by appending an explanation to the task name. Examples for good task names:

- deliver order
- deliver large order
- deliver large order with truck

Gateways
Gateways only need to be named if they are divergent exclusive Gateways. Divergent exclusive gateways should consist of at least a noun followed by a verb followed by a question mark to indicate what is evaluated at this gateway. Alternatively a whole questions can be asked. Examples for good gateway labels:

- order delivered?
- was order delivered in time?

Sequence Flows
Sequence flows only need to be named if they come out if diverging Event based, exclusive or inclusive gateways. Sequence flows that follow one of those gateways should have labels that indicate the outcome of the gateways condition.

General Rules
All Tasks, Events and Gateways should have a label. Sequence flows should have a label if they leave a exclusive, inclusive or event based gateway. Generally the naming convention should be in it consistent and for readability it is recommended to avoid long names. As a rule of thumb, having labels that are no longer than 5 words is recommended[11].

CHAPTER 3

Evaluating and Optimizing Executable Process Models

3.1 Six Sigma Approaches

The first set of analysation and optimization techniques discussed in this chapter originate from the Six Sigma initiative. The name Six Sigma originates from the interval of 6σ in the normal distribution that indicates the aimed success rate of 99.99966% [21][26] . A representation of the statistical meaning can be see in figure 3.1. Apart from the goal to decrease the error rate, 6σ is also an methodology for systematically improving process quality [24].

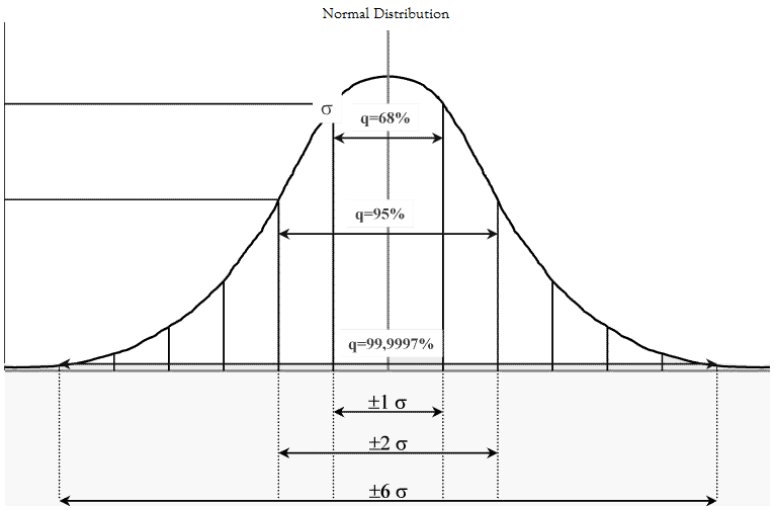


Figure 3.1: The standard normal distribution showing the 6σ interval (graphic from [26])

Applied in the context of (executable) business processes, Six Sigma provides a set of methods to identify and eliminate inefficient or needless steps in a process [27]. While countless tools are available as part of Six Sigma, the two main methodologies applied in these tools are **D**efine, **M**easure, **A**nalyze, **I**mprove, **C**ontrol [20] (DMAIC), used for improving existing processes, and **D**efine, **M**easure, **A**nalyze, **D**esign, **V**erify [20] (DMADV), used when creating new processes [20] .

In the following, a few selected Six Sigma methods are described that apply the **DMAIC** and **DMADV** approach, these being:

- SIPOC Analysis
- Value Added Analysis

To demonstrate a few of the following methodologies, the exemplary *Missing Part Process* will be used. The fictional process demonstrates what happens when a customer of a furniture shop reports a missing part in his/her order and requests a new part to be shipped. The corresponding BPMN-model is shown in Figure 3.2

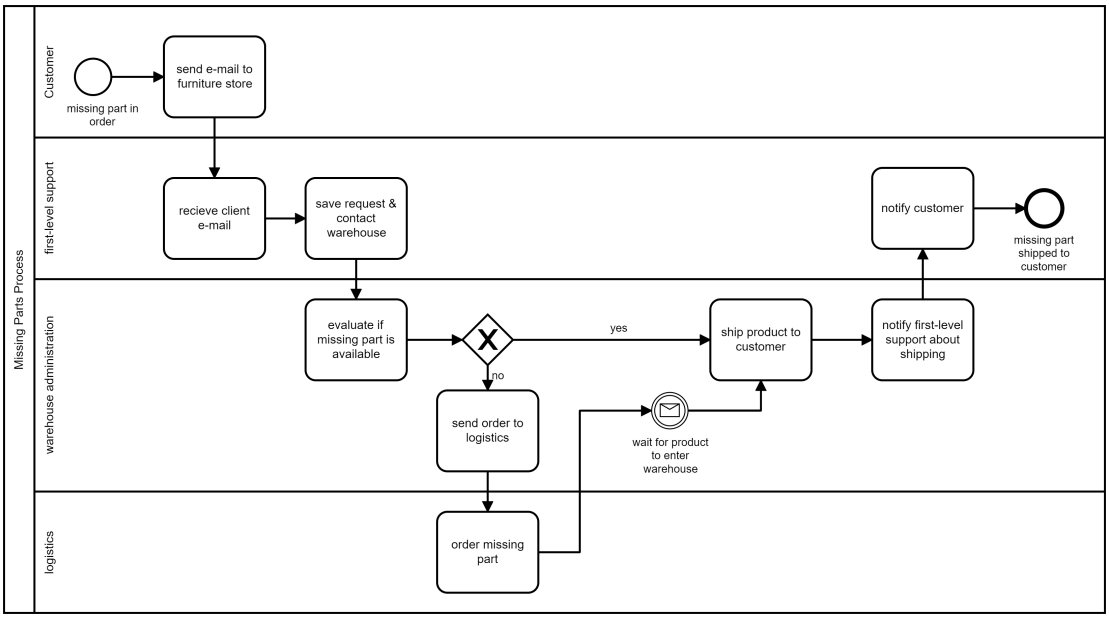


Figure 3.2: The BPMN model for the *Missing Part Process*

3.1.1 SIPOC Analysis

The SIPOC analysis is a tool used in the **Define** phase of DMAIC and DMADV and should provide more detailed information about each process step than the BPMN model [27].

The outcome of a SIPOC analysis is a table, containing the following information for each task in the process [25]:

- **S**upplier: provides the input for this task and influence the output the supplier can be internal or external.
- **I**nput: Resources, data or material needed for this task provided by the suppliers.
- **P**rocess: the series of actions taken in this task. This can also be a subprocess.
- **O**utput: represent the result of this process task and provide value for the customer
- **C**ustomer: the customer that benefits from the output of this task (can be external or internal of the organization)

When this method is applied to the **Missing Part Process** the resulting SIPOC table looks like this:

Supplier	Input	Process	Output	Customer
customer	The specification for the missing part, unique order number	send e-mail to furniture store	e-mail containing: The specification for the missing part, unique order number	first-level support
customer	The specification for the missing part, unique order number	recieve client e-mail	knowledge about the missing part	first-level support
first-level support	The specification for the missing part	save request & contact warehouse	database entry for missing part	warehouse administration
warehouse administration	The specification for the missing part	evaluate if missing part is available	if the missing part is available or not	warehouse administration
warehouse administration	The specification for the missing part	send order to logistics	filled in order for missing part	logistics
logistics	The specification for the missing part	order missing part	confirmation for order	warehouse administration
warehouse administration	customer address, packaged missing part	ship product to customer	packaged product	warehouse administration
warehouse administration	expected shipping date, unique order number	notify first-level support about shipping	e-mail containing: expected shipping date, unique order number	first-level support
first-level support	expected shipping date, unique order number	notify customer	e-mail containing: expected shipping date, unique order number	customer

While the SIPOC analysis is no tool for optimizing a process itself, it is necessary in order to have an overview of all activities in a process for other Six Sigma techniques.

3.1.2 Value Added Analysis (VAA)

The aim of value-Added analysis is to determine the value of each activity in a process and find possible steps in a process that can be eliminated [11][16]. When performing value-added analysis every step in a process is classified by the kind of value it adds [16]:

- **RVA (Real value-added)**: Steps that contribute to the output the external customer is expecting are viewed as *real value adding*.

- **BVA (Business value-added)**: Some steps in processes are not performed with the customer in mind but are needed to keep the company running. These are classified as *business value adding*.

- **NVA (No value-added)**: Steps that neither contribute to external customer requirements nor keep the business running are considered as *non value adding*. There are two kinds of such steps, activities that are necessary because of bad process design (e.g. reworking, waiting, moving,...) and activities of bureaucratic nature (e.g. reporting, logging, ...).

To make this more practical, H. James Harrington provided a flowchart 3.3 with which every activity can be classified in one of those three categories[16].

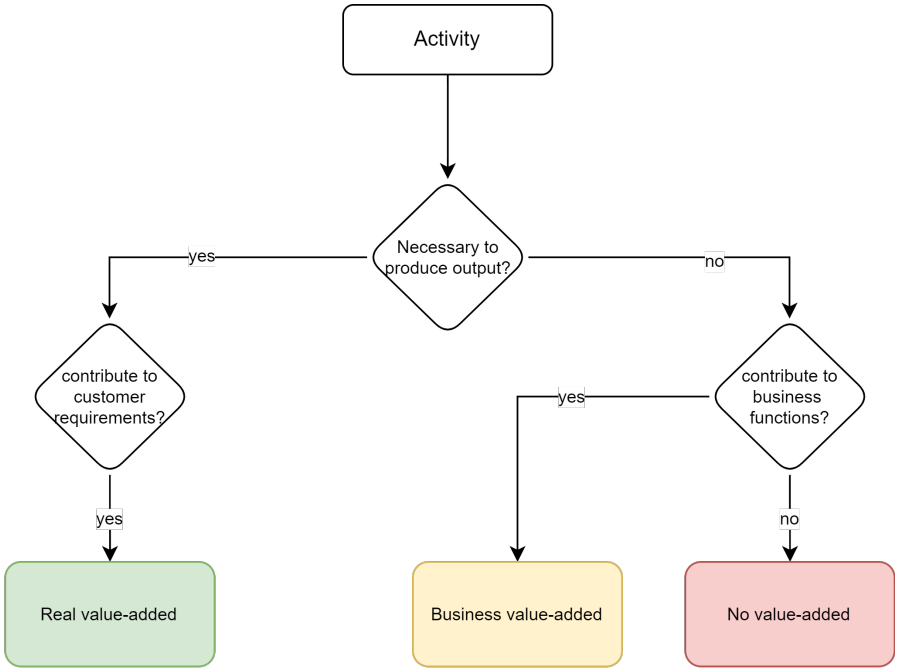


Figure 3.3: The value added analysis flowchart adapted from [16]

In figure 3.4 the tasks in the missing parts process (figure 3.2) are colored according to their category after applying the flowchart in figure 3.3. Green tasks are real value adding activities, yellow tasks business value adding activities and red tasks are no value adding activities.

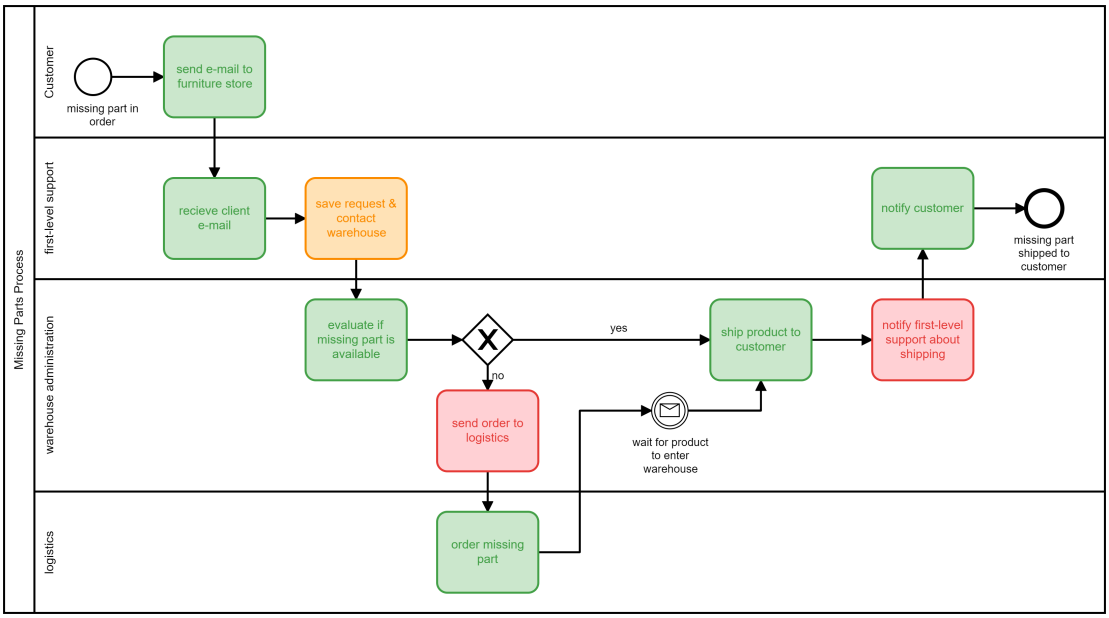


Figure 3.4: missing parts process (figure 3.2) with tasks colored based on their value

After this categorization is done, one can start with waste elmination, emaning eliminating unnecesarry steps in the process. The obvious goal is to eliminate NVA steps. This an be done by automation. In the context of the missing parts process (figure 3.2) this could mean automating the customer notification after the ordered product enters the warehouse. Another approach for waste elimination is minimizing handovers. For example could the missing parts process be changed in a way, that the warehouse administration directly triggers the product ordering instead of sending the order to the logistics department. Apart from eliminating NVA steps, this is also the opportunity to evaluate BVA steps. If the BVA steps do not align actual business goals or do not bring enough value to the business goal they are applied for, they can be eliminated or simplified. [11][16]

3.2 Quantitative Analysis

The issue with qualitative methods for process improvement is the inability to quantify the impact of their application. In order to measure and justify changes in the status quo process we need quantitative methods for analyzing processes. In the context of Business Process Management (BPM), assessable measures about three different dimensions of a process can be investigated : The process as a whole, resources involved in the process and individual Tasks in a process. [11]

Before discussing methods for measuring process performance, it is necessary to define what is measured. The four key performance measures examined in this chapter are: cost, time, quality and flexibility. [11]

The following chapter will present methods for qualitative analysis of processes. Starting with the definition of the four dimensions of process performance: cost, time, quality and flexibility. Afterwards the following methods for measuring performance will be described:

- Flow Analysis
- Queues
- Simulation

3.2.1 Performance Measures

As mentioned above, the performance of a process can be measured using the four dimensions: cost, time, quality and flexibility.

Cost
The purpose of Process Redesign is often to reduce cost. Another perspective could also be to increase Turnover, yield or revenue of the process. A cost notion that is often from interest when it comes to process redesign is operational cost which also includes labor cost. Automation is often viewed as the solution when it comes to reducing labor cost, the costs that are needed for developing and maintaining a software should not be neglected [11]. Introducing a (BPMN) Workflow Management System itself is quite expensive and scales for some systems with the number of processes that are executed.

Time
Cycle time or *throughput time* is the time it takes for a process from start to finish [8]. In the case of the *Missing Part Process* shown in 3.2, the cycle time would be the time that elapses from the moment the customer notices that a part is missing to the time the missing Part is shipped. While the *Cycle time* of the whole process as a metric is highly relevant for client satisfaction, it is on its own usually not tangible when it comes to improvement of the process. The *cycle time* usually consists of two constituents [11]:

- *Processing time*: The time where the request is actively processed, meaning the process participants (e.g. people or software) actually spend this time handling the process.
- *Waiting time*: The time a process spends waiting. Waiting time includes the time the process is waiting in a queue for resources to finish the processing of previous processes and other waiting times, like waiting for the product to enter the warehouse in the *Missing Part Process*

Quality

Flexibility

3.2.2 Flow Analysis

The Idea behind (work)flow analysis is to estimate the overall performance of a process given the performance of individual tasks.

Flow analysis is especially useful in order to analyze alternative process designs. By considering the performance of each step in the process, the benefit of process redesigns can be quantified and the impact of automating individual tasks becomes visible.

In the following the analyzed performance measure that is looked at will be execution time.

Sequential Tasks
The first calculation starts with sequential tasks or blocks. The execution time of consecutive tasks in a process can be added together to calculate the total time of those two tasks. Considering the tasks in 3.5 where the runtime of each task is brackets below the task name, this would result in a process time of 30s for this block. [15] [11]

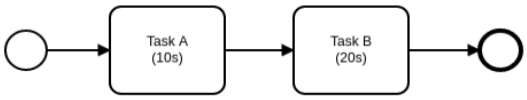


Figure 3.5: a process with two sequential executed tasks

Considering a more general process model P with sequential blocks $B_1, B_2...B_n$ the runtime R_P of process P ca be calculated using the formula 3.1.

$$R_P = \sum_{i=1}^{n} B_i \tag{3.1}$$

Parallel Tasks
The cycle time of parallel tasks or blocks can be calculated using the maximum time of the two parallel blocks. Given the parallel block in 3.6, this would result in a process time of 20s for this block.[11]

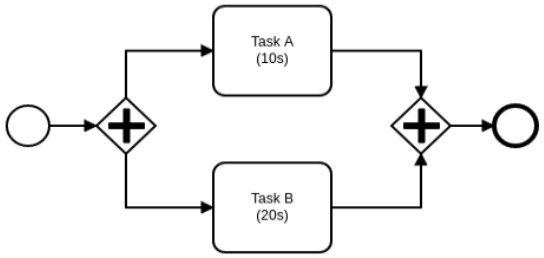


Figure 3.6: a process with two parallel executed tasks

Considering a more general process model P with parallel blocks $B_1, B_2...B_n$ the runtime R_P of process P ca be calculated using the formula 3.2.

$$R_P = \arg\max_i B_i \tag{3.2}$$

Alternative Sequence Flows
When having two or more alternative sequence flows with different execution times, additional knowledge about the likelihood of those alternatives is needed [11]. When looked at the alternative executed tasks in 3.7, the average cycle time can be estimated by multiplying the probability of the two alternatives with the execution time of the tasks and summing up the results. This results in $10 * 70\% + 20 * 30\% = 13s$ estimated execution time.

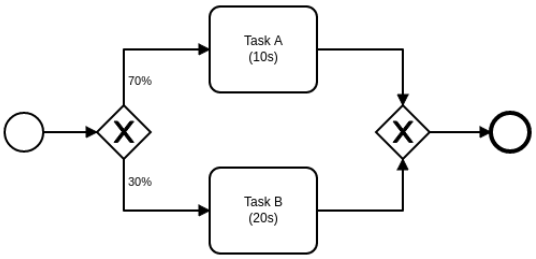


Figure 3.7: a process with two alternative sequence flows

Generally, the cycle time R_P of a process P with one or more alternative blocks, that have the runtimes $B_1, B_2...B_n$ and a likelihood of $p_1, p_2...p_n$ respectively, is estimated with the formula 3.3. [11]

$$R_P = \sum_{i=1}^{n} B_i * p_i \tag{3.3}$$

Repeating Tasks
According to the BPMN-Standard[18] defined by OMG, there are multiple ways to indicate that a task or sequence is executed more than once:

- **Multiple Instances**: Three lines in the bottom of the task indicate that a task is executed multiple times - once for every instance or element. There are two kinds of muli-instance tasks, sequential, meaning the instances are processes one after another and parallel, meaning the instances are processed at the same time.

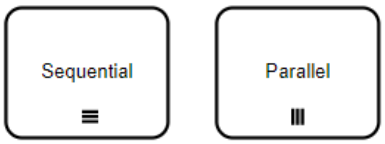


Figure 3.8: Multiple Instance Tasks

The execution time of these tasks can be estimated using the same calculations as described above with parallel and sequential tasks.

Having a Process P consisting of only one multi-instance task with the runtimes $B_1, B_2...B_n$ for each instance, the total runtime R_P of process P ca be calculated using the formula 3.2 when the instances are executed parallel and 3.1 when the instances are executed sequential.

- **Activity Looping**: A loop in the bottom center of the activity indicates, that this task si performed more than once. The definition on when this task repeats is specified in the *loopCharacteristics*-XML Element.

Figure 3.9: A Task with Activity Looping

- **Sequence Flow Looping**: Whenever a whole sequence flow instead of a task needs to be repeated, a *Sequence Flow Looping*-pattern can be used where the looping condition is defined within a exclusive gateway. The default flow points back to the start of the repeating sequence flow.

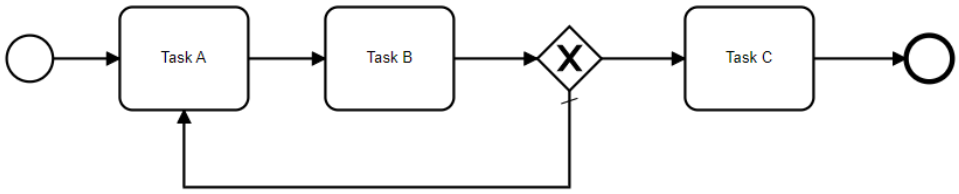


Figure 3.10: The Sequence Flow Looping Pattern

The execution time of both **Sequence Flow Looping** sequences and **Activity Looping** tasks are dependent on the probability that these sequences are repeated. Looking at the two equivalent processes in 3.11 the probability of Task A repeating is 20% for each loop taken. Task A is always executed at least once. The probability for Task A being executed twice is $20\% = 0.2$. The probability of Task A being executed three times is $0.2 * 0.2 = 0.04$. Following this pattern, the execution time can be estimated with the infinite sum $\sum_{i=1}^{\infty} 0.2^i * 10s$.

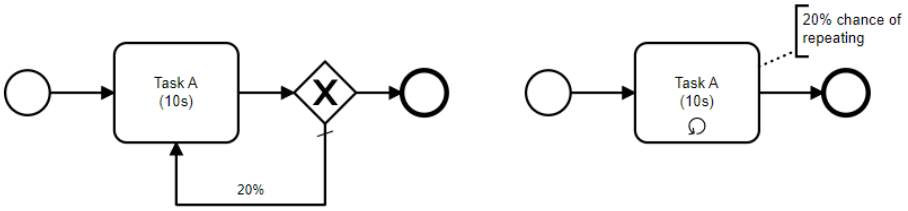


Figure 3.11: a process with sequence flow looping (left) and the equivalent using activity looping (right)

Generally a the total execution time R_P of a repeated sequence where the inner part has a execution time of $R_{Repeated}$ and a probability of $p, 0 < p < 1$ to be repeated in each loop can be calculated using the formula int 3.4

$$R_P = \sum_{i=1}^{\infty} p^i * R_{Repeated} \tag{3.4}$$

Queues As mentioned before, the cycle time of a process consists of the time the process is actively executed (*processing time*) and the time the process spends waiting (*waiting time*). In the previous section basic flow analysis was introduced to approximate and analyze these measures. While the processing time can be approximated using basic flow analysis, getting an idea about the total expected waiting time of a process is more complex, since flow analysis as it is has no tool to estimate queuing time.

In order to bridge this gap, the flow analysis can be extended using an queuing approximation algorithm [15].

3.2.3 Simulation

3.2.4 Cycle Time Efficiency & Littles Law

As mentioned before, the cycle time (CT) of a process consists of the time the process is actively executed (*processing time*) and the time the process spends waiting (*waiting time*). Besides looking at overall cycle time, it might also be useful to analyze the processing time relative to the cycle time. This measure is called *cycle time efficiency*. In this context, the *processing time* is often also called *theoretical cycle time (TCT)*. The *cycle time efficiency* (CTE) can be calculated using the formula 3.5.[11]

$$CTE = \frac{TCT}{CT} \tag{3.5}$$

The theoretical cycle (TCT) time can be calculated by applying flow analysis. For determining the cycle time (CT) two other measures are needed.

The first one is the *arrival rate (λ)* that denotes the average number of process instances that start the process in a given period. The second measure needed is the *work-in-process (WIP)* which is the average number of active process instances at any given time. If the WIP is stable, meaning it does not increase infinitely, the Process itself is called stable. Those three measures, the arrival rate λ, the WIP and the cycle time (CT) are related through *Littles Law* which is described in formula 3.6.[17]

$$WIP = \lambda * CT \tag{3.6}$$

Usually, the actual cycle time of a process is harder to determine than the WIP or the arrival time, therefore one can transform *Littles Law* to determine the cycle time using the other two measures (see formula 3.7).

$$CT = \frac{WIP}{\lambda} \tag{3.7}$$

CHAPTER 4

Concept and Implementation

The following chapter will describe the software that was implemented as part of this thesis. The software is supposed to read in an BPMN2.0 Diagram and return a set of suggestions how this BPMN can be improved according to the best-practices described in the first chapters.

This chapter will start with describing the technical implementation and used technologies that were used as well as details on how the software is structured and how it can be used. Moreover, this chapter will describe the REST-interface that was implemented and Objects returned by the software.

Finally the suggestions that can be made to an executable BPMN will be described in section 4.3. Some of this suggestions will not be suitable to automation. Explanations on why certain suggestions where implemented and others not will also be provided in this section.

4.1 Software Architecture

The Software was implemented as an Spring Boot Server[9] Application. The software was developed using git[6] as version control system and the git project is available as an open source project on `https://github.com/dsunaric/epms-service`.

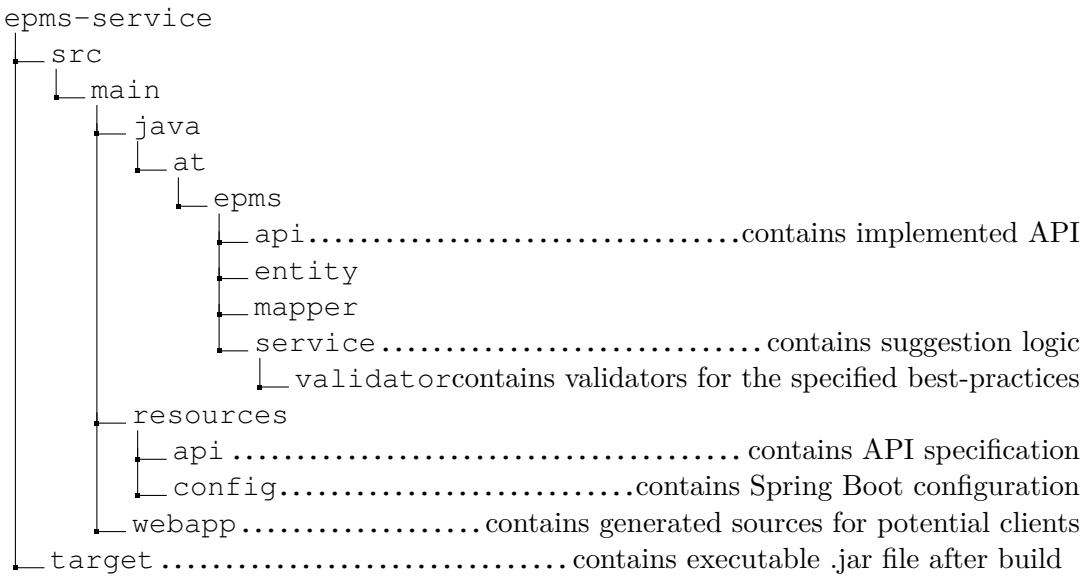
This software is implemented using maven[7] as a build-tool, therefore software artifacts can be added to the local repository using the command `mvn install`. This also generates the API sources into the `target` directory. After that the packaged .jar file can be run using the command `java -jar target/*.jar`.

4.1.1 BPMN Processing

For reading in and processing the BPMN models the Camunda-Model-API[4] was used. The reason for that was the detailed documentation and better usability compared to using an XML parser. The Camunda Model API is able to process most BPMN 2.0 elements. A list of supported elements can be found in the *instance* package[5].

4.1.2 Folder structure

The following visualizes and describes the important parts of the projects folder structure:



4.2 Interface

The API was designed and developed using OpenAPI and Swagger[10]. The OpenAPI definition is written in one *.yaml* file which is located in the folder `epms-service/src/main/resources/api`.

After Starting the Software, the API description is accessible on `localhost:8080/documentation/v3/api-docs` and can directly be used for code generation by potential frontend applications.

The application has only one REST-Endpoint, `GET /suggestions`. This endpoint has the process model as request body and returns a list of **AppliedRules** Objects.

One **AppliedRule** object has the following attributes:

- **title**: The title of the applied Rule
- **description**: The description of the rule. Explains when the Rule applies.

- **details**: Explains details about the rule and gives explanation about the effected elements.
- **effectedElements**: A list of elements that violate this rule. The id, name and type (event or task) of the effected element is returned.

4.3 Suggestions for improvement

This section describes a set of possible suggestions for improving a BPMN Model. Not all of those rules where implemented as some are not suitable for automation or would be beyond the scope of this thesis.

4.3.1 Comply with Naming Conventions

BPMN models, no matter if conceptual or executable, should meet naming conventions as it is described in section 2.3.6. One aspect of naming conventions is the recommended maximum length of five words for any label.

Since natural language processing would go beyond the scope of this thesis and naming conventions apply to not only executable BPMN, this suggestion for improvement was not fully automated in the context of this thesis. Because of its simplicity, suggesting to rename a element when the label is too long (greater than five words) was implemented.

4.3.2 Extend automation boundaries

The first step of turning an conceptual process model executable is identifying the automation boundaries 2.3.1. In order to optimize na existing process one could think of extending those existing boundaries and brainstorming ideas to automatize tasks that are not yet automatable.

This requires a lot of domain specific knowledge and can therefore not be simply implemented.

4.3.3 Eliminate Manual Tasks

As describes in section 2.3.2 Manual Tasks should not be part of an executable BPMN diagram and should if possible be automated using service tasks or, if that is not possible, be replaced by user tasks.

Manual tasks are identified by the software. The returned effected elements represent a list of manual tasks in the given diagram.

4.3.4 Complete the process model

Another manual step to improve a process model, is making sure the process model is complete section 2.3.3.

Since this would also require domain specific knowledge, the software does not include analyzing the process model for incompleteness.

4.3.5 No two consecutive Tasks handled by the same resource

Two task that are executed one after the other should be merged, if they have the same enitity executing the task. As described in section 2.3.4, bringing the model to an adequate granularity level minimizes handovers and overhead. In case of User tasks this means the same usergroup is executing both tasks. Automated Tasks that follow each other should also always be merged to minimize flownodes.

The software recognizes such consective tasks return a list of effected elements that represent the tasks that can be merged with its direct successor. In case that three or more tasks can be merged, this algorithm will return every task that can be merged with its successor on its own. Therefore if “task1“ , “task2“ and “task3“ can be merged all together, the algorithm will return “task2“ and “task3“ as effected elements.

An example on when this rule is satisfied can be found in figure 4.1. The software will return *Task A*, Task B and *Task D* as effected elements, indicating, that *Task D* should be merged with its successor (*Task E*), *Task A* should be merged with *Task C* and *Task B* should be merged with *Task C*, therefore merging all three Tasks (*Task A*, *Task B* and *Task C*)together. An example for an resulting process model, after this suggestions are applied, can be found in figure 4.2.

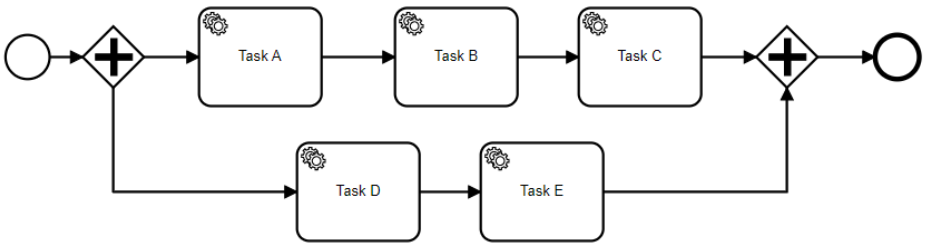


Figure 4.1: Example of a process where tasks can be merged together

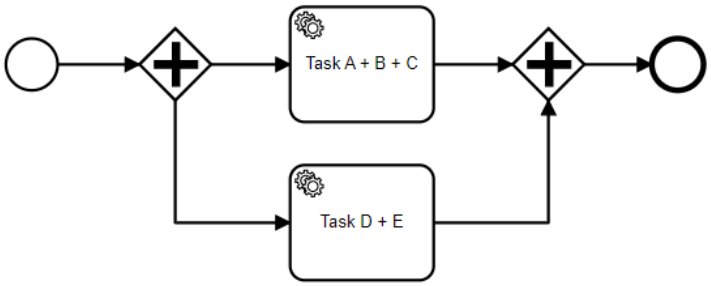


Figure 4.2: change necessary for the process in 4.1 so satisfy this rule

4.3.6 Inclusive Gateways over combining parallel and exclusive Gateways

In order to save flownodes, inclusive gateways should be used instead of a parallel gateway followed by one or more exclusive gateways. An example for this kind of pattern is shown in 4.3.

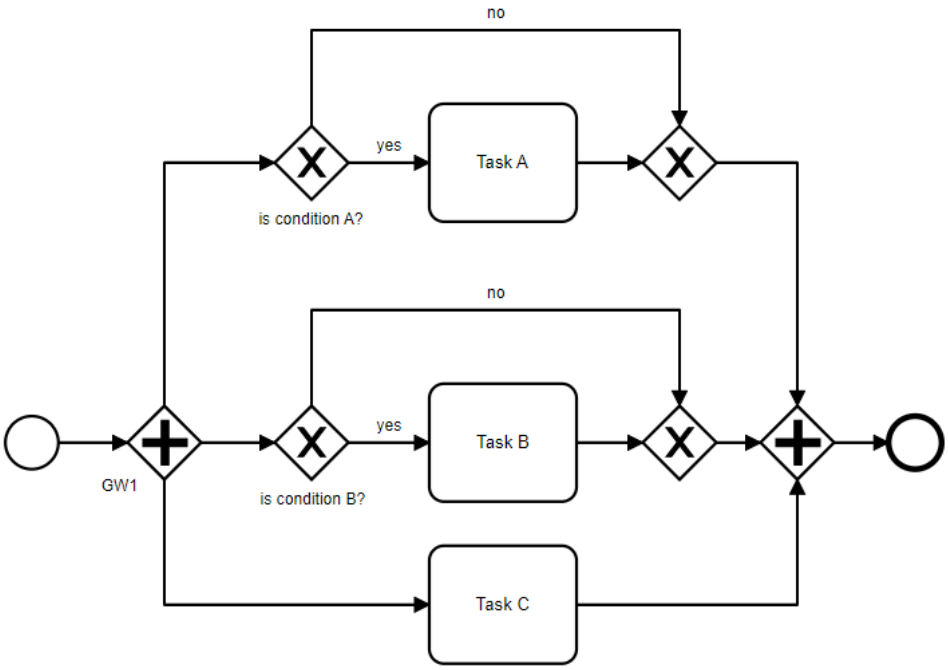


Figure 4.3: Example for a parallel gateway followed by one or more exclusive gateways

The software applies this rule whenever a parallel gateways is followed by one or more exclusive gateways. The returned effected elements are a list of parallel gateways that are followed by one or more exclusive gateways in the given diagram. In the example

shown in 4.3, the returned element would be *GW1* and the change to comply with this rule would be shown in 4.4

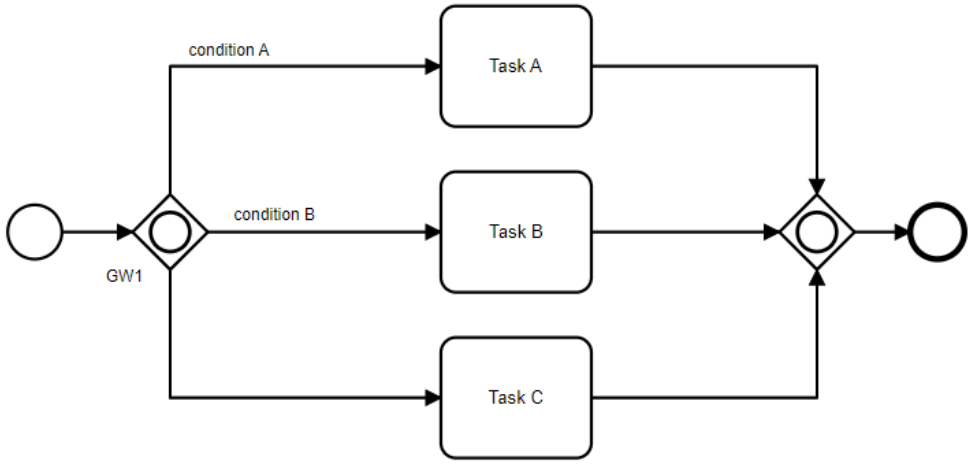


Figure 4.4: change necessary for the process in 4.3 so satisfy this rule

4.3.7 Value Added Analysis

In order to eliminate tasks in a process that have limited value for the customer, a value added analysis as explained in section 3.1.2 can be performed.

This would also require domain specific knowledge and is therefore not done by the software.

4.3.8 Evaluate Suggestions Quantitative

Finally, all process models as a result of applying any of those suggestions made by the software or manually, can be evaluated against each other using quantitative measures. The section 3.2 describes approaches to estimating quantitative measures using flow analysis.

Since BPMN 2.0 has no option to integrate estimations for quantitative measures for processes or tasks, and estimation would require detailed knowledge about the process at hand, this will not be implemented.

CHAPTER 5

Case Study

In the previous chapter a set of suggestions for improving a BPMN diagram where listed. As it was also mentioned, only a fraction of these improvements can be automated.

This chapter will provide a example how the mentioned not automatable and automatable suggestions can be applied together to an existing BPMN model.

5.1 The example model

The used BPMN model for this chapter will be

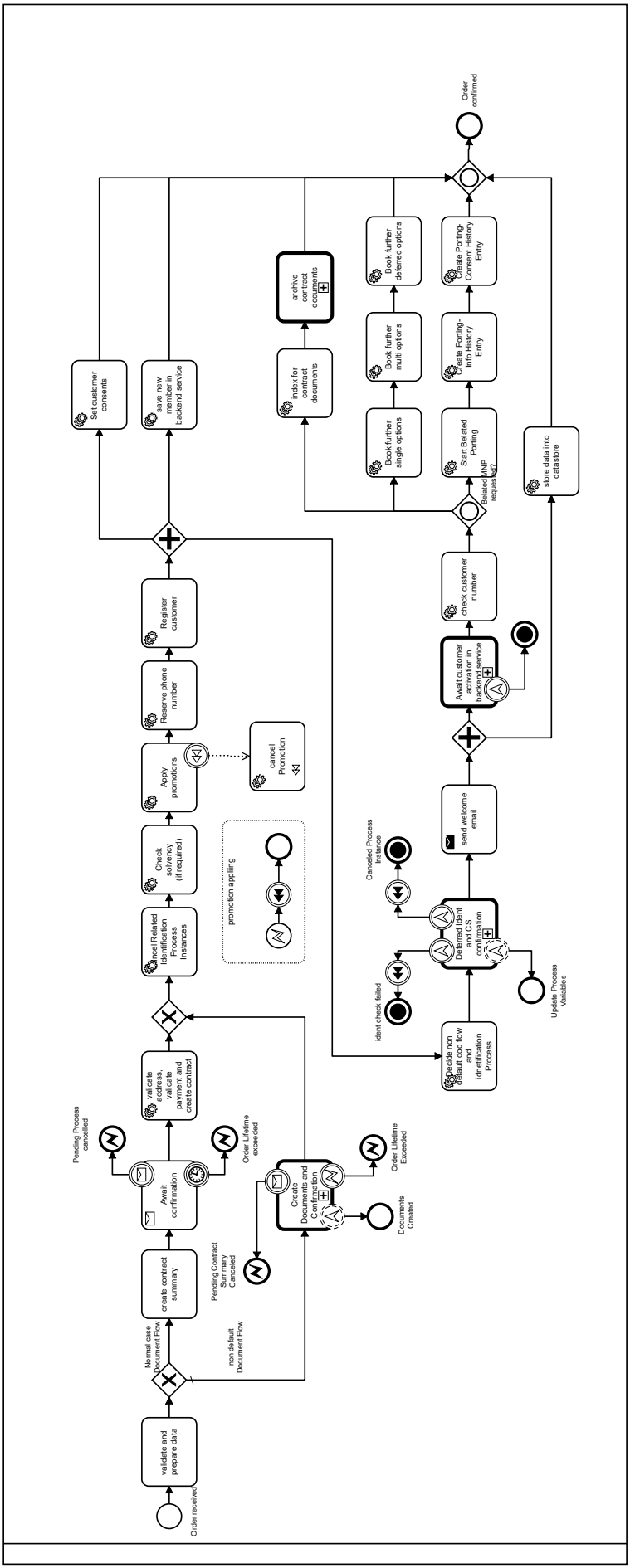


Figure 5.1: Example of a process where tasks can be merged together

5.2 Comply with Naming Conventions

5.2.1 renaming tasks

5.2.2 renaming gateways

5.2.3 rename events

5.3 Extend automation boundaries

5.4 Eliminate Manual Tasks

5.5 Complete the process model

5.6 No two consecutive Tasks handled by the same resource

5.7 Inclusive Gateways over combining parallel and exclusive Gateways

5.8 Value Added Analysis

5.9 Evaluate Suggestions Quantitative

CHAPTER 6

Conclusion

write conclusion

List of Figures

List of Tables

Listings

Acronyms

BPMN Business Process Model and Notation. 3–8, 23

BPMS Business Process Management System Business Process Management System. 3, 4

XML Extensible Markup Language XML. 8

XSD XML Schema Definition XSD. 8

Bibliography

[1] Activity customers. https://www.alfresco.com/customers. Accessed: 2022-04-21.

[2] Bpmn xml schema specification. https://www.omg.org/spec/BPMN/20100501/BPMN20.xsd. Accessed: 2022-04-13.

[3] Camunda customers. https://camunda.com/about/customers/. Accessed: 2022-04-21.

[4] Camunda model api. https://docs.camunda.org/manual/7.16/user-guide/model-api/. Accessed: 2022-06-26.

[5] Camunda model api - supported elements. https://docs.camunda.org/javadoc/camunda-bpm-platform/7.16/org/camunda/bpm/model/bpmn/instance/package-summary.html. Accessed: 2022-06-26.

[6] Git. https://git-scm.com/. Accessed: 2022-06-26.

[7] Maven. https://maven.apache.org/. Accessed: 2022-06-26.

[8] Six sigma terms. https://www.sixsigmadaily.com/six-sigma-terms/. Accessed: 2022-04-26.

[9] Spring boot. https://spring.io/projects/spring-boot. Accessed: 2022-06-26.

[10] Swagger. https://swagger.io/. Accessed: 2022-06-26.

[11] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management*. Springer Publishing Company, Incorporated, 2nd edition, 2018.

[12] J. Freund and B. Rücker. *Real-Life BPMN (4th Edition): Includes an Introduction to DMN*. Independently Published, 2019.

[13] Andreas Gadatsch. *Grundkurs Geschäftsprozess-Management: Analyse, Modellierung, Optimierung und Controlling von Prozessen*. Springer-Verlag, 2020.

[14] Horst Gruber and Christian Huemer. Profitability analysis of workflow management systems. In *2009 IEEE conference on commerce and enterprise computing*, pages 233–238. IEEE, 2009.

[15] Byung-Hyun Ha, Hajo A Reijers, Joonsoo Bae, and Hyerim Bae. An approximate analysis of expected cycle time in business process execution. In *International Conference on Business Process Management*, pages 65–74. Springer, 2006.

[16] H James Harrington. Value analysis (value-added analysis). In *The Innovation Tools Handbook, Volume 2*, pages 377–394. Productivity Press, 2016.

[17] John DC Little and Stephen C Graves. Little's law. In *Building intuition*, pages 81–100. Springer, 2008.

[18] OMG. Business Process Model and Notation (BPMN), Version 2.0.2, December 2013.

[19] Razvan Radulian. Rethinking bpmn: how to build good process models using bpmn, 2020.

[20] K Selvi and Rana Majumdar. Six sigma-overview of dmaic and dmadv. *International Journal of Innovative Science and Modern Engineering*, 2(5):16–19, 2014.

[21] Samia M Siha and Germaine H Saad. Business process improvement: empirical assessment and extensions. *Business process management journal*, 2008.

[22] Geno Stefanov. Business process automation with bpms. In *Proceedings of International Conference on Application of Information and Communication Technology and Statistics in Economy and Education (ICAICTSEE)*, page 617. International Conference on Application of Information and Communication ..., 2014.

[23] Gerardo Navarro Suarez, Jakob Freund, and Matthias Schrepfer. Best practice guidelines for bpmn 2.0. *BPMN*, 2, 2010.

[24] Geoff Tennant. *Six Sigma: SPC and TQM in manufacturing and services*. Routledge, 2017.

[25] Helge Toutenburg and Philipp Knöfel. *Six Sigma: Methoden und Statistik für die Praxis*. Springer-Verlag, 2008.

[26] N Vivekananthamoorthy and S Sankar. Lean six sigma. In *Six Sigma Projects and Personal Experiences*. IntechOpen, 2011.

[27] Jan Vom Brocke and Michael Rosemann. *Handbook on business process management 1: Introduction, methods, and information systems*. Springer, 2014.