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, 13th July, 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, 13th July, 2022

Dragana Sunaric

Acknowledgements

write Acknowledgements

Abstract

BPMN is a commonly used diagramming language to visualize business processes. Executable process models are an effective tool for bridging the gap between IT experts, that are responsible for process automation, and domain experts, who have insight knowledge about the process at hand. This thesis explains best practices for modeling and transforming executable BPMN process models to make these expressive and readable.

Based on this best practices, a list of suggestions that can be used to improve an existing BPMN model was formulated.

Since some best practices are candidates for automation, a software was implemented as part of this thesis that can scan a BPMN model for violations of mentioned best practices. This software is freely available as an open source project on github: `https://github.com/dsunaric/epms-service`.

This thesis goes then on to demonstrate in a case study how the software can be used as an aid to implement the suggestions. The process used for this case study originated from a real live client registration process used in the telecommunication sector and was adapted to be used in this thesis.

Contents

CHAPTER 1

Introduction

Today Workflow Managment System (WfMS)s like Camunda BPM and Alfresco Process Services (BPM) are widely used by big organizations [**?**] [**?**]. Real-life processes being executed in WfMSs can be complex and have to be changed and reevaluated together with the software that is executed in these processes as requirements and guidelines for the specific domain change continuous.

Using an WfMS to directly automate processes defined as Business Process Model and Notation (BPMN) has the advantage to be readable for IT experts as well as domain experts. As a consequence both parties have a common understanding of the process and communication is easier when changes have to be made to the process or the software. 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 a 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 nonnecessary 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 to satisfy these best practices.

1.2 Structure of This Thesis

This goal will be achieved by stating the current research on creating executable BPMN models in chapter 2.

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. To compare two process models, this work will also provide methods for measuring metrics like time and cost of processes defined as BPMN models. This will be shown in chapter 3

Along with this thesis, a software was developed that scans a given BPMN model for used Anti-patterns. The documentation and implementation details of the developed software can be found in 4.

Finally, this thesis will provide detailed information about how to use mentioned guidelines together with the developed software in practice with a case study in chapter 5.

CHAPTER 2

Automating Business Processes

The following chapter will provide an overview of the current research on best practices when creating 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 to open the possibilities of automating parts of a process or even the whole process. While process automation in itself can be useful, there are some other advantages of introducing an WfMS, besides the possibility to automate certain process steps.

2.1.1 Shorten Process Lifetime

Managing processes manually requires handling tasks such as handling handovers from one entity to another. This 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. [**?**]

2.1.2 Reduce Process Cost

By reducing process lifetimes and increasing productivity due to better handling of resources process costs can be reduced [**?**]. in this context, it has to be said, that due to the high price for workflow management systems the overall cost does not have to decrease necessarily after implementing an WfMS [**?**].

2.1.3 Workload Reduction

As stated earlier, managing processes needs to be done either by an WfMS or manually which creates an additional workload for an organization. Workloads for employees executing the processes are also kept steady due to dynamic resource assignments. [**?**][**?**]

2.1.4 Enforce Rules

Defining the processes that are directly executed and controlled by an WfMS enables the organization to enforce the execution of the process as it is designed without giving employees room for shortcuts. Following rules and protocols regarding the process can be automated and enforcing guidelines and laws in an organization becomes easier. [**?**]

2.1.5 Create Transparency

Using an WfMS provides insights into the actual processes that are executed in the organization. This makes it easier to determine the performance of processes by providing historical information of completed process instances and provides insight into the current status of processes that are still in progress. [**?**]

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 experts [**?**]. They are typically 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 [**?**].

Due to their incompleteness, business-oriented models cannot be executed in an WfMS just like that 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 readable 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 by an WfMS. Besides the visual information, executable BPMN model also need to contain execution properties like interface definitions and variables that are used by the process. Those variables are called Process Variables.[**?**]

An algorithm on how to efficiently turn an conceptual BPMN model executable as stated in the book *Fundamentals of Business Process Management* [**?**] 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 to how this conversion can take place. In [**?**] performing such a transformation is broken down into 5 steps:

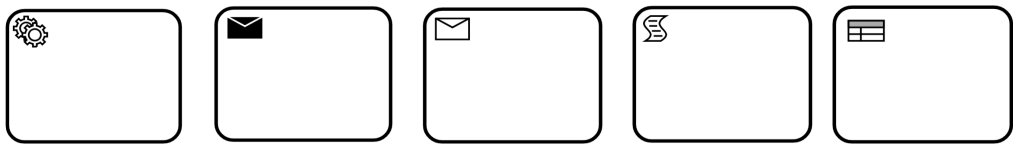
1. Identify the automation boundaries
2. Review manual tasks
3. Complete the process model
4. Bring the process model to an adequate granularity level
5. Specify execution properties

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 WfMS.

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

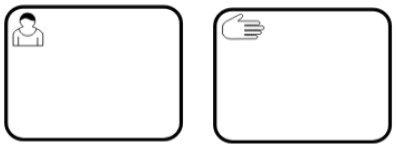
- **Service Task**: A Task that invokes a service. This service can be a webservice or application code.
- **Send Task**: Used to send a message to an external participant (A participant that is not part of the process).
- **Receive Task**: Used to receive a message from an external participant.
- **Script Task**: Executes a script that can be interpreted by the WfMS.
- **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 [**?**]

Usually, not every step in a process can be fully automated. Processes can also have **manual tasks** and **user tasks**. While both describe activities performed by humans,**user tasks** are aided by the orchestration and resource management functionalities of a WfMS and **manual tasks** do not interact with the business process execution engine at all.



(a) User Task (b) Manual Task

Figure 2.2: Manual and user tasks according to the BPMN 2.0 standard [**?**]

2.3.2 Review Manual Tasks

As mentioned earlier manual tasks are not automated and do also happen without the aid of a WfMS. Most WfMS implement manual tasks as a pass-through activity, meaning the manual task is ignored [**?**][**?**][**?**]. To incorporate all aspects of the process into the WfMS, it is necessary to evaluate if the identified manual tasks in our BPMN-model can be substituted by other BPMN constructs. This can be done in the following ways:

- **Automate the task**: Trying to extend the current automation boundaries would be the best case scenario Receive Task. Depending on the nature of the task that is currently performed manually, this might not always be possible.
- **Turn it into a user task**: If the task cannot be automated or the organization lacks the resources to automate the task yet, another possibility is to change the manual task into a user task. The WfMS manages task assignments and after completion, the system is notified via a worklist handler. [**?**]

In the case that neither an automated task nor a user task is suitable for modeling the manual task, one might also consider isolating the task and modeling 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 considered to be executed using a WfMS[**?**, p. 228]

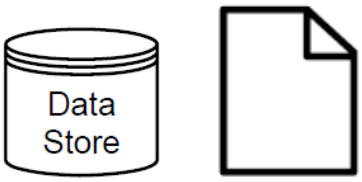
2.3.3 Complete the Process Model

Usually, conceptual BPMN models are not complete and leave out certain information that is seen as implicit knowledge or as not important by the person modeling the process. These gaps in the model might be crucial when a complete picture of the process is needed for automation.

A common incompleteness in many conceptual models is ignoring errors and only implementing the 'happy-path' of a process. 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 model, the corresponding executable BPMN model has to define what happens in case any error occurs.

Apart from defining alternative process paths, in this step of turning a model executable, it is also necessary to model the input and output data of tasks and gateways. In the BPMN 2.0 specification this is done 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 [**?**]

2.3.4 Bring the Process Model to an Adequate Granularity Level

The granularity of tasks in an conceptual BPMN model does differ from the granularity needed in an executable BPMN model. As mentioned before, using an WfMS is not just about the possibility of automation, but mainly the orchestration and deciding who needs to be doing which task. [**?**]

Therefore consecutive tasks that are done by the same participant should be clustered together to minimize handovers that have to be unnecessarily processed by the workflow management system. [**?**] However, there are some exceptions to this rule:

- **Tracking progress**: To know how much the process has advanced, it can be useful to split certain tasks even if they are done by the same participant.
- **Handling exceptions**: If different errors and exceptions can occur for a set of tasks that is performed by the same participant, it is recommended to keep these tasks separated.
- **Managing resources**: Sometimes consecutive tasks are performed by participants that have the same role, but could be done by two different participants to maximize capacity. In this case, it is also recommended 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 implementation details of our BPMN model. While the changes performed up to this step had an impact of the graphical rendering, the execution properties are not graphically embedded into BPMN but are encoded into the Extensible Markup Language XML

(XML) specification of the BPMN-model. [?] A schematic representation of the structure of BPMN is provided in figure 2.4. For an insight in the full specification of the BPMN 2.0 XML, the XML-schema can be found on the OMG-Website[?].

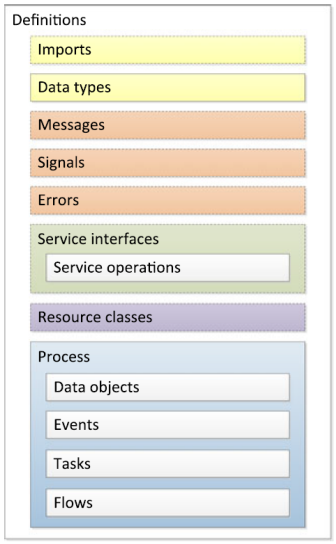


Figure 2.4: Structure of the BPMN format [?]

Process Variables

To use data in different elements of our process, we need process variables that can be read, created, and modified during the process's 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 as shown in listing 2.1.

```
1   <xs:element name="person">
2   <xs:complexType>
3   <xs:sequence>
4   <xs:element name="name" type="xs:string"/>
5   <xs:element name="address" type="xs:string"/>
6   <xs:element name="city" type="xs:string"/>
7   <xs:element name="country" type="xs:string"/>
8   </xs:sequence>
9   </xs:complexType>
10  </xs:element>
```

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

The corresponding complex XML object can be seen in listing 2.2.

```
1   <person>
2   <name>Dragana Sunaric</name>
3   <address>Wiedner-Hauptstrasse 5</address>
```

```
4         <city>1040 Wien</city>
5         <country>Austria</country>
6         </person>
```

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

The definition of common errors, messages, and escalations that are either thrown or listened to by events and tasks are also part of the execution properties.

Messages
Every BPMN element has at least an **id** that identifies the given element and a descriptive **name** of the element.[**?**] Despite the properties that all BPMN elements have, messages have no additional variables that need to be set. An example of a message specification can be seen in listing 2.3.

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

Listing 2.3: Example for a message definition

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. An example error is defined in listing 2.4.

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

Listing 2.4: Example for a error definition

Escalations
Similar to errors, escalations additionally have an **escalationCode** that can be used to listened to this escalation by events. An example for a escalation specification is shown in listing 2.5.

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

Listing 2.5: Example for a escalation definition

Input and Output Variables
The earlier mentioned process variables are active during the whole process life-cycle. While having globally available variables has its advantages, some data used in the process might only be used in single tasks. There are even approaches to eliminate globally available process variables altogether [**?**]. For data that is only used in individual

elements of the model, it is possible to define input and output values for each task. These variables are only visible within the specific task and have to be defined as an XSD-schema file, similarly to complex process variables. [**?**]

Service Tasks
Service tasks can be used to call external applications or web services. To do that, the interaction with the given service has to be defined in the process model. The called service has to provide an interface that describes the available service operations and their parameters as well as the associated return values. Service operations can be synchronous, meaning the process instance waits for the operation 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 WfMS does this by copying the above-mentioned Input values of the Task into the service call and if defined, copying the output values of the service call into the output values of the service task. [**?**]

2.3.6 Apply Naming Conventions

To create readable models, it is recommended to apply naming conventions for BPMN 2.0 diagrams. While there are many suggestions for naming conventions, none is defined or recommended by the BPMN 2.0 standard [**?**] itself. The following is a suggestion for naming different BPMN elements based on [**?**], [**?**] [**?**] and [**?**].

Events
Names of events should start with a business object (noun) followed by a verb. 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 of 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 of 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 and a question mark to indicate what is evaluated at this gateway. Alternatively, a whole question can be asked. Examples of good gateway labels:

- *order delivered?*
- *was order delivered in time?*

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

General Rules
All tasks, events, and gateways should have a label. Sequence flows should have a label if they leave an 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[**?**].

CHAPTER 3

Evaluating and Optimizing Executable Process Models

This chapter will give an introduction to how existing process models can be evaluated and optimized. First, it will show how process steps can be eliminated using *value added analysis* as part of the *six sigma* initiative. After that, this chapter will give an introduction to methods for measuring metrics like time and cost of processes defined as BPMN models to better evaluate two alternative processes.

3.1 Six Sigma and Value Added Analysis

3.1.1 Introduction to Six Sigma

The first analyzation and optimization technique discussed in this chapter originates from the six sigma initiative. The name six sigma stands for the interval of 6σ in the normal distribution that indicates the aimed success rate of 99.99966% of a process [**?**][**?**]. A representation of the statistical meaning can be seen in figure 3.1. Apart from the goal to decrease the error rate, 6σ is also a methodology for systematically improving other aspects of process quality [**?**].

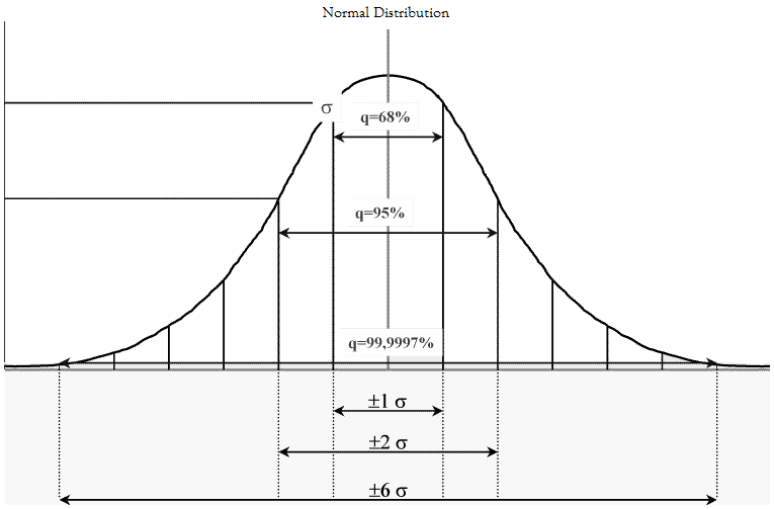


Figure 3.1: The standard normal distribution showing the 6σ interval (graphic from [?])

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 [?] (DMAIC), used for improving existing processes, and **D**efine, **M**easure, **A**nalyze, **D**esign, **V**erify [?] (DMADV), used when creating new processes [?].

Applied in the context of (executable) business processes, six sigma provides a method to identify and eliminate inefficient or needless steps in a process, called *value added analysis* [?].

3.1.2 Value Added Analysis (VAA)

The aim of the value added analysis is to determine the value of each activity in a process and identify activities that can be eliminated [?][?]. When performing value added analysis, every step in a process is classified by the kind of value it adds [?]:

- **RVA (Real value added)**: Activities that contribute to the external customers expected output are viewed as *real value adding*.
- **BVA (Business value added)**: Some activities 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)**: Activities 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 3.2 with which every activity can be classified into one of that three categories[**?**].

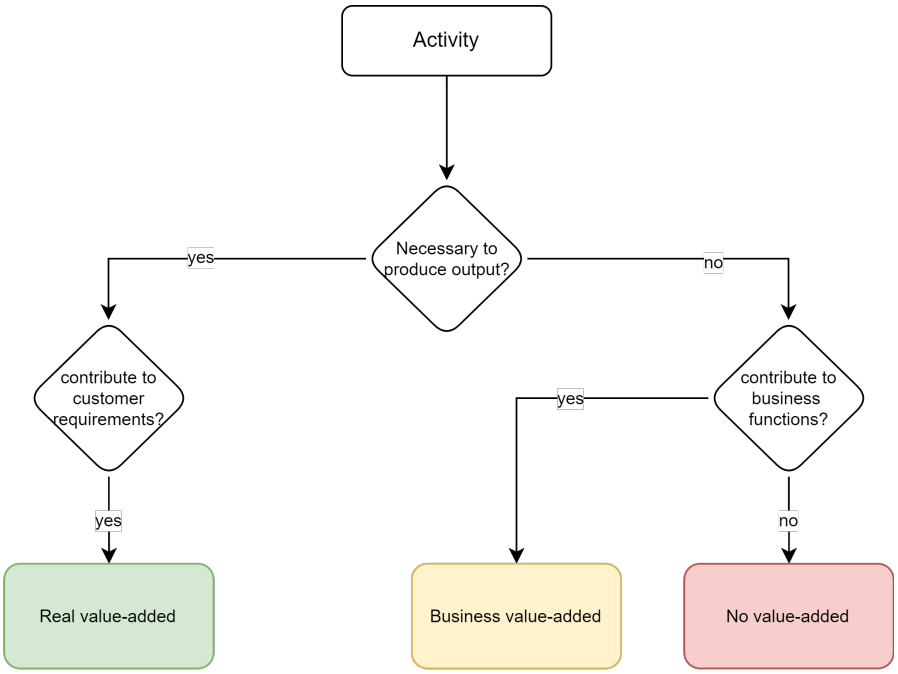


Figure 3.2: The value added analysis flowchart adapted from [**?**]

To demonstrate the value added analysis, the exemplary *Missing Part Process* will be used. This 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 process model is shown in figure 3.3

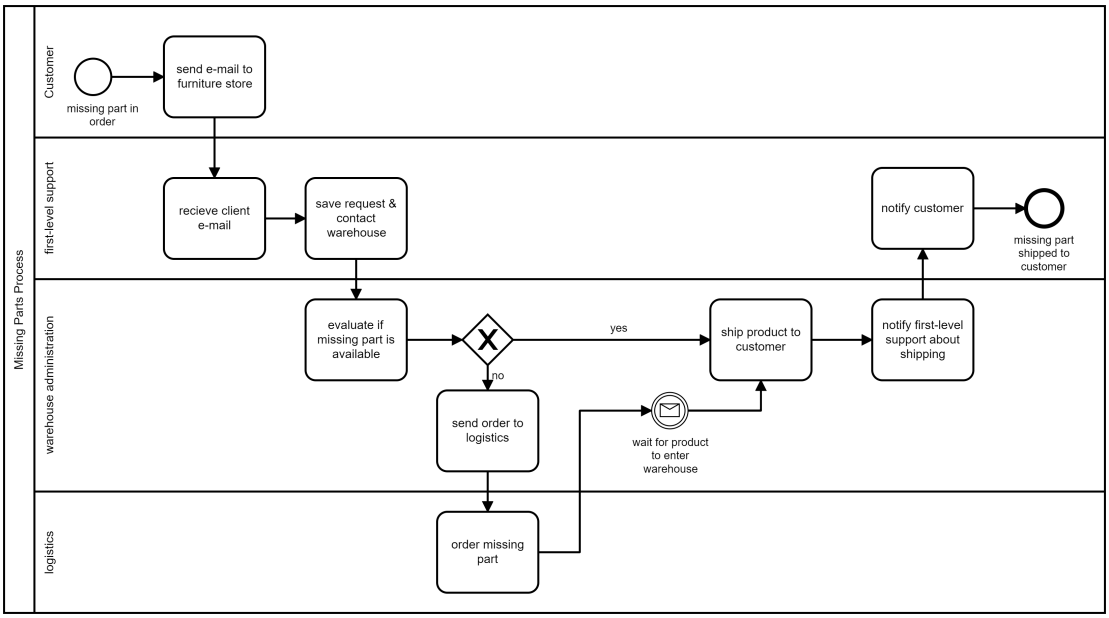


Figure 3.3: The *Missing Part Process* BPMN model

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

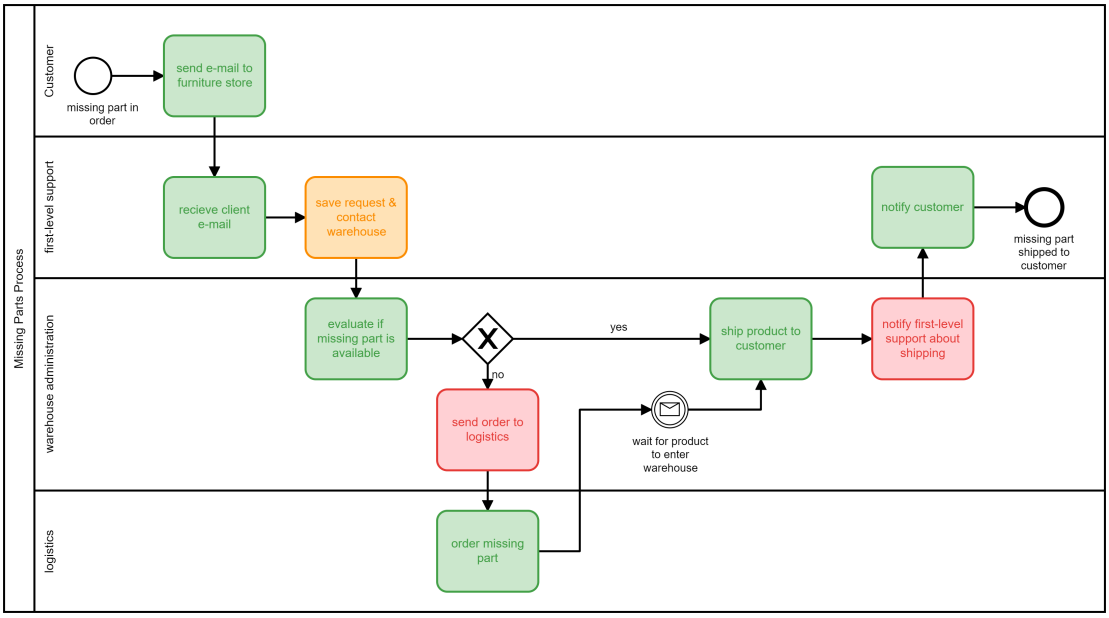


Figure 3.4: missing parts process (figure 3.3) with tasks colored based on their value

After this categorization is done, one can start with waste elimination. The first goal is to try to eliminate the impact of NVA activities. This can be done for example by automating the activity. In the context of the missing parts process (figure 3.3) this could mean automating the customer notification after the ordered product enters the warehouse.

Another approach for waste elimination is minimizing handovers. For example, the process can be altered 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 with actual business goals or do not bring enough value to the business goal they are applied for, they should also be eliminated or simplified. [**?**][**?**]

3.2 Quantitative Analysis

To measure and justify changes in the status quo process, a quantitative method for analyzing and comparing alternative processes is needed. In the context of Business Process Management (BPM), accessible 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. [**?**]

Before discussing methods for measuring process performance, it is necessary to define what is measured. The three key performance measures of a process are: cost, quality, and time[**?**]

The following chapter will present methods for qualitative analysis of processes. Starting with the definition of the three dimensions of process performance.

3.2.1 Performance Measures

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

Cost
The purpose of process redesign is often to reduce cost or to increase the turnover, yield, or revenue of the process. A cost notion that is often of 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 costs but the costs that are needed for developing and maintaining software should not be neglected [**?**]. Introducing a WfMS itself is quite expensive and scales for some systems with the number of processes that are executed.

Quality
Another metric important for a process is the quality of the process output. One quality

measure is the error rate. According to the six sigma initiative, the goal should be to have an error rate below 99.99966%.

Time
Cycle time or *throughput time* is the time it takes for a process from start to finish [**?**]. In the case of the *missing part process* shown in 3.3, 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 the improvement of the process. The *cycle time* usually consists of two constituents [**?**]:

- *Processing time*: The time where the request is actively processed, meaning the process participants (e.g. people or software) spend this time executing activities.
- *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 an event to happen.

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 to analyze alternative process designs. By considering the performance of each step in the process, the potential benefit of process redesigns can be quantified and the impact of automating individual tasks becomes visible.

In the following section, the analyzed performance measure that is used for evaluation 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 figure 3.5, where the runtime of each task is brackets below the task name, this would result in a processing time of 30s for this block. [**?**] [**?**]

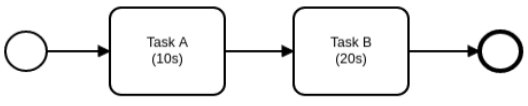


Figure 3.5: a process with two sequential executed tasks

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 figure 3.6, this would result in a process time of 20s for this block.[**?**]

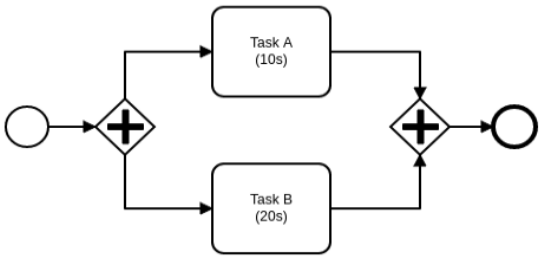


Figure 3.6: a process with two parallel executed tasks

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 [**?**]. When looking at the alternative executed tasks in figure 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.

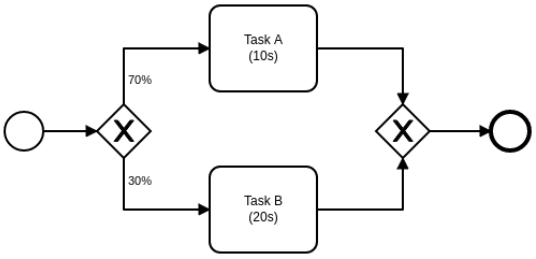


Figure 3.7: a process with two alternative sequence flows

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. [**?**]

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

Repeating Tasks
According to the BPMN-Standard[**?**] 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 multi-instance tasks: sequential, meaning the instances are processes one after another and parallel, meaning the instances are processed at the same time.

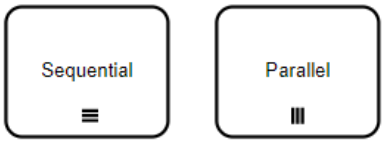


Figure 3.8: Multiple instance tasks

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 is 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.

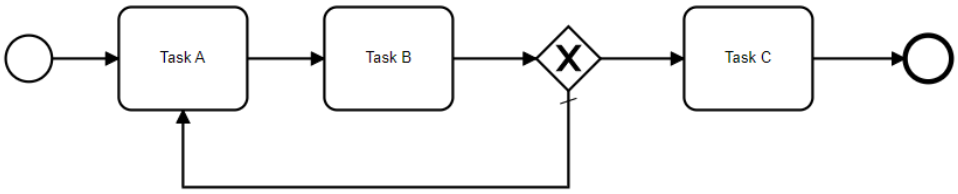


Figure 3.10: The sequence flow looping pattern

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 $\sum_{i=1}^{\infty} 0.2^i * 10s$.

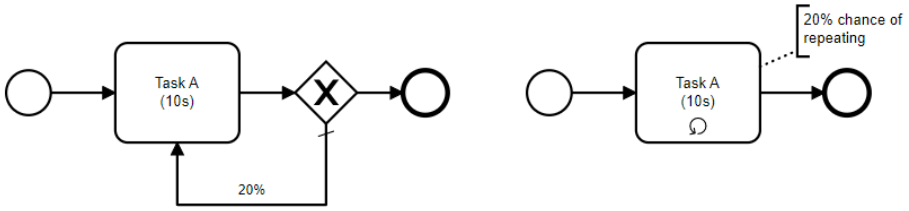


Figure 3.11: a process with sequence flow looping (left) and the equivalent using activity looping (right)

Generally the total execution time R_P of a repeated sequence where the inner part has an 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 has no tool to estimate queuing time.

To bridge this gap, the flow analysis can be extended using a queuing approximation algorithm [**?**].

3.2.3 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 the 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.[**?**]

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

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.[**?**]

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

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 on how this BPMN can be improved.

This chapter will start by 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 the 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 these suggestions will not be suitable for automation. Explanations on why certain suggestions were implemented and others not will also be provided in this section.

4.1 Software Architecture

The software was implemented as a spring boot server[**?**] application. The software was developed using git[**?**] as a version control system and the git project is available as an open source project on `https://github.com/dsunaric/epms-service`.

The implemented uses maven[**?**] 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[**?**] was used. The reason for that was the detailed documentation and better usability compared to using an XML parser. The Camunda Model API can process most BPMN 2.0 elements. A list of supported elements can be found in the *instance* package[**?**].

4.1.2 Folder structure

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

```
epms-service
├── src
│   └── main
│       ├── java
│       │   └── at
│       │       └── epms
│       │           ├── api ..................... contains implemented API
│       │           ├── entity
│       │           ├── mapper
│       │           └── service ................. contains suggestion logic
│       │               └── validator contains validators for the specified best-practices
│       ├── resources
│       │   ├── api ............................. contains API specification
│       │   └── config .......................... contains Spring Boot configuration
│       └── webapp .......... contains generated sources for potential clients
└── target ..................... contains executable .jar file after build
```

4.2 Interface

The API was designed and developed using OpenAPI and Swagger[**?**]. 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 affected elements.
- **affectedElements**: A list of elements that violate this rule. The id, name, and type (event or task) of the affected element are returned.

4.3 Suggestions for improvement

This section describes a set of possible suggestions for improving a BPMN Model. Not all of those rules were 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 renaming an element when the label is too long (greater than five words) was implemented.

4.3.2 Extend automation boundaries

The first step of turning a conceptual process model executable is identifying the automation boundaries 2.3.1. To optimize an 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 described 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 affected 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 in 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 Merge consecutive tasks handled by the same resource

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

The software recognizes such consecutive tasks and returns a list of affected elements that represent the tasks that can be merged with its direct successor. In case 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 affected 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 affected 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.

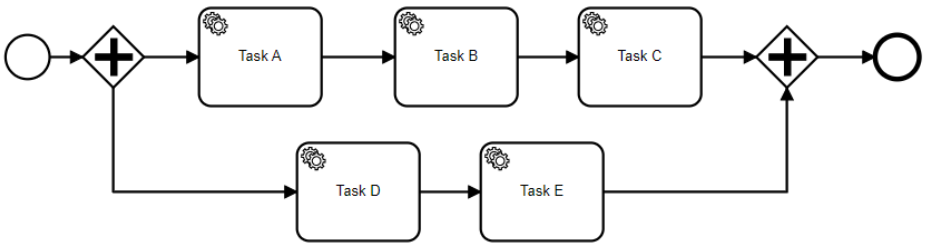


Figure 4.1: Example of a process where tasks can be merged

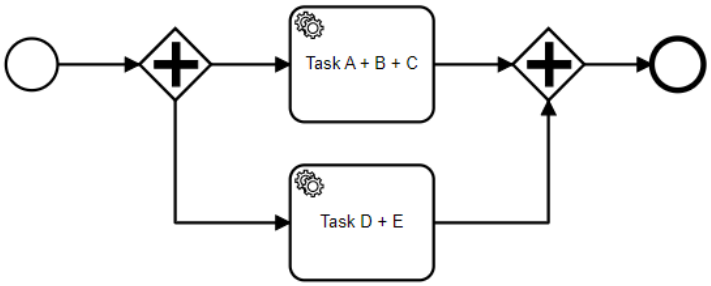


Figure 4.2: change necessary for the process in 4.1 so satisfy this rule

Inclusive Gateways over combining parallel and exclusive Gateways

The following suggestion does not originate directly from the best practices mentioned in chapter 2 and chapter 3 but is a common anti-pattern when designing processes. To save flow nodes, inclusive gateways should be used instead of a parallel gateway followed by one or more exclusive gateways. An example of this kind of pattern is shown in 4.3.

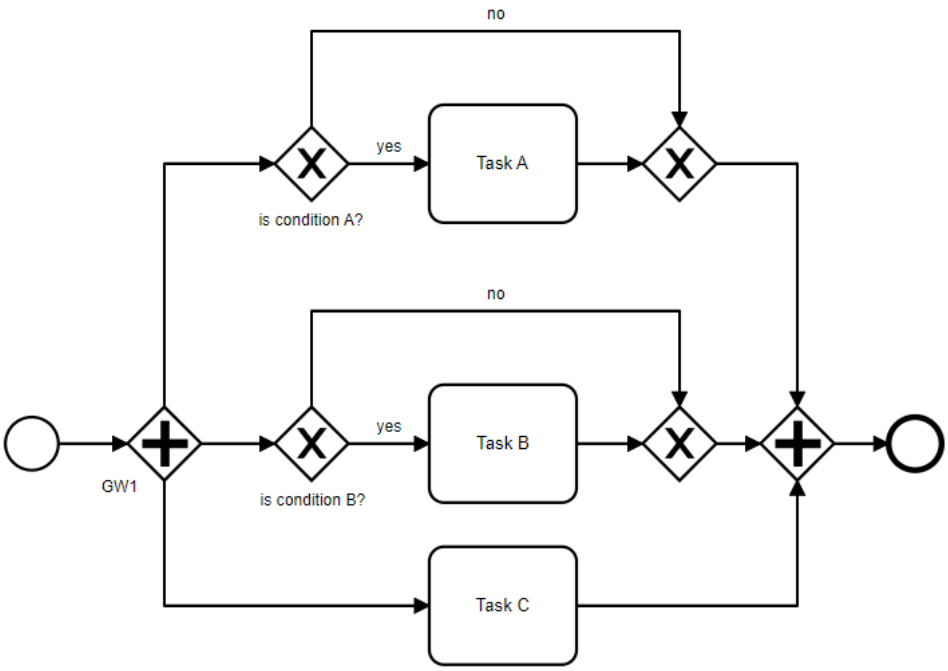


Figure 4.3: Example for a parallel gateway followed by one or more exclusive gateways

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

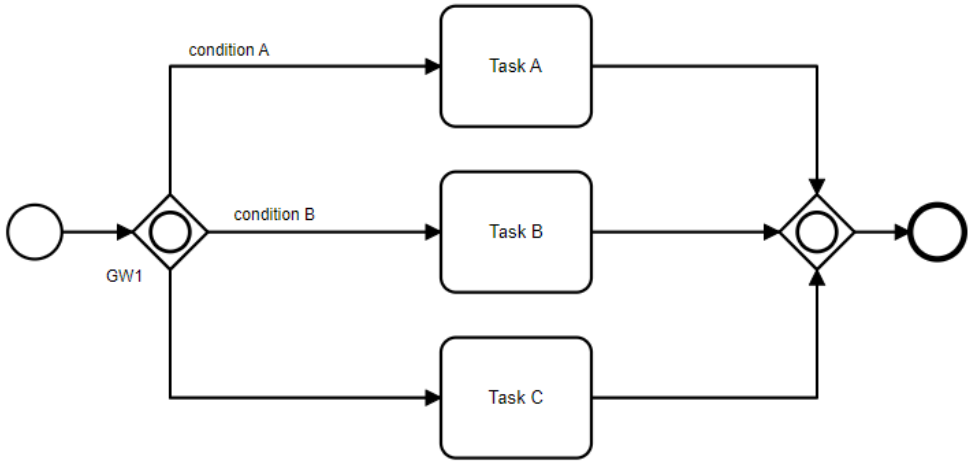


Figure 4.4: change necessary for the process in 4.3 so satisfy this rule

4.3.6 Perform Value Added Analysis and Waste Elimination

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.

4.3.7 Evaluate Suggestions Quantitative

Finally, all process models that result from 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 was listed. As was also mentioned, not all of these improvements can be automated.

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

At first, the example BPMN model used for this case study and the evaluation that is done by the software on this BPMN will be explained. Finally, all suggestions as they are listed in section 4.3 will be implemented one by one, using the software evaluation as an aid.

5.1 The Example Model

The used BPMN model for this chapter will be a fictitious client registration process that was adapted from a real live process used in the telecommunication sector to be used in this thesis. The graphical representation of the process model can be found in figure 5.1. The full *.bpmn* file of this process can be accessed in the GitHub repository[**?**].

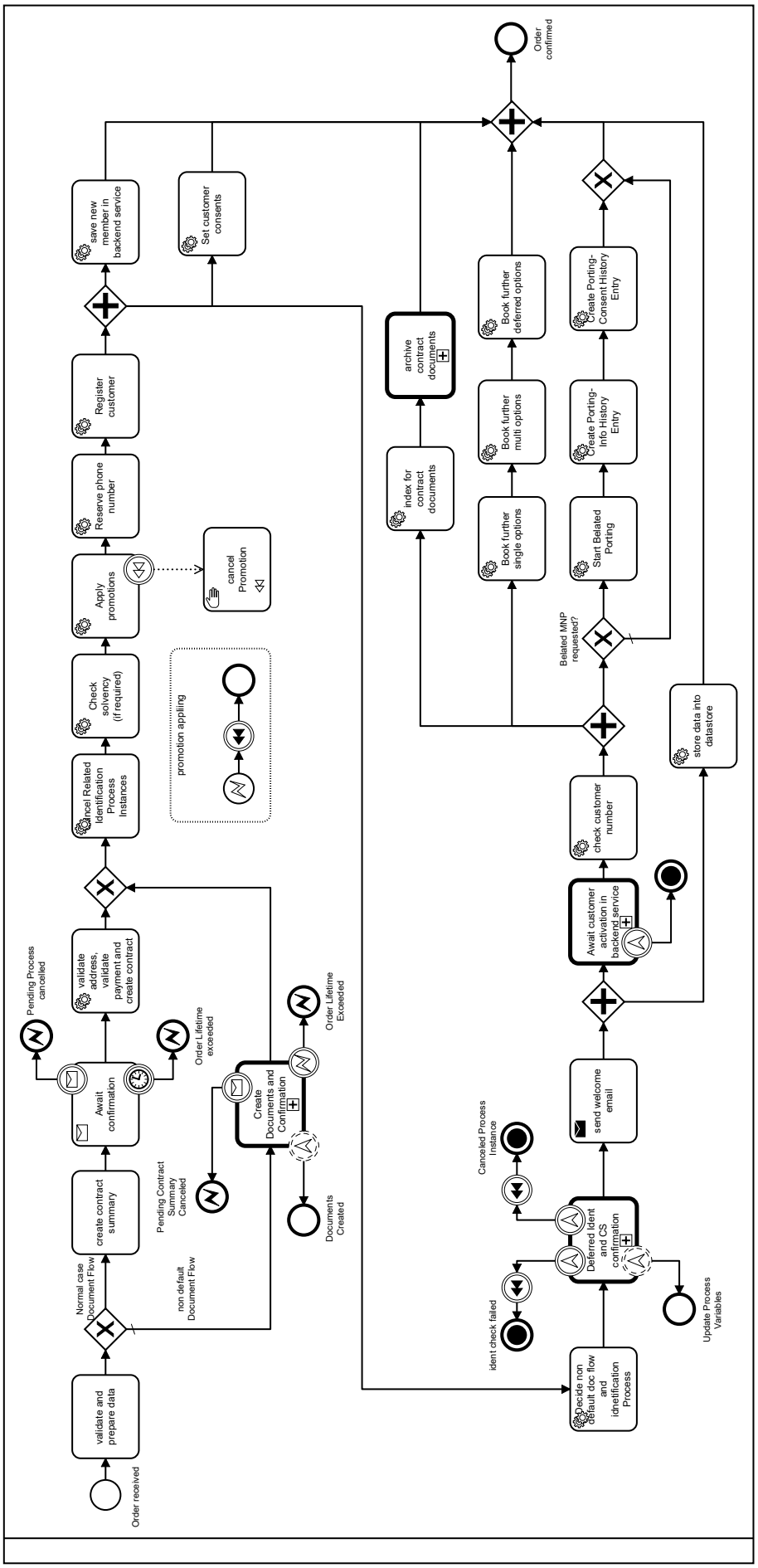


Figure 5.1: Adapted customer registrations process

5.2 Evaluation by the Software

To evaluate the best practices that were implemented the BPMN Model was given as input for the software. The following output for each suggestion/rule was given. The full output can be found in the appendix section 6.

Comply with Naming Conventions

As described in chapter 4.3.1, this algorithm scans the BPMN model for events, tasks, or gateways that have more than five words in their label. The following list of elements is returned that violate this rule:


```
"affectedElements": [
{
        "id": "
           ServiceTask_DefaultDocWorkflowPostAwaitConfirmation
           ",
        "name": "validate address, validate payment and
           create contract",
        "type": "Task"
},
{
        "id": "DecideIdAndCsProcess",
        "name": "Decide non default doc flow and
           idnetification Process",
        "type": "Task"
},
{
        "id": "save_member_for_riskident",
        "name": "save new member in backend service",
        "type": "Task"
}
```


Eliminate Manual Tasks

When searching for manual tasks in the process (see section 4.3.3), only one can be identified and is returned:


```
"affectedElements": [
{
        "id": "data_to_warehouse",
        "name": "store data into datastore",
        "type": "Task"
}
```


Merge Consecutive Tasks Handled by the Same Resource

Another rule that was implemented was that no two consecutive tasks should be processed by the same resource. The software identified the following tasks that could be merged with its successor:

```
"affectedElements": [
{
        "id": "CancelRelatedWebIdent",
        "name": "Cancel Related Identification Process
           Instances",
        "type": "Task"
},
{
        "id": "ReserveMsisdn",
        "name": "Reserve phone number",
        "type": "Task"
},
{
        "id": "BookAutoTopups",
        "name": "Book further multi options",
        "type": "Task"
},
{
        "id": "ServiceTask_0p0tajd",
        "name": "Start Belated Porting",
        "type": "Task"
},
{
        "id": "ServiceTask_0e2r82f",
        "name": "Create Porting-Info History Entry",
        "type": "Task"
},
{
        "id": "BookSingleTopups",
        "name": "Book further single options",
        "type": "Task"
}
]
```


Replace Combinations of Parallel and Exclusive Gateways With Inclusive Gateways

As described in section 4.3.5 it is preferred to use inclusive gateways instead of combining parallel and exclusive gateways. The only element that violates that rule in the given BPMN is returned:


```
"affectedElements": [
{
        "id": "ParallelGateway_0crudor",
        "name": null,
        "type": "Gateway"
}
]
```

5.3 Apply Suggestions for Improvement

The following section will give an introduction on how the suggestions mentioned in section 4.3 can be applied to the process in figure 5.1.

5.3.1 Comply with Naming Conventions

Based on the naming conventions defined in section 2.3.6, unnamed tasks, events, and gateways were named and elements returned in listing 5.2 were renamed. In figure 5.2 a segment of the example BPMN, where the naming conventions are violated, is shown, and figure 5.3 shows the corresponding changes after renaming.

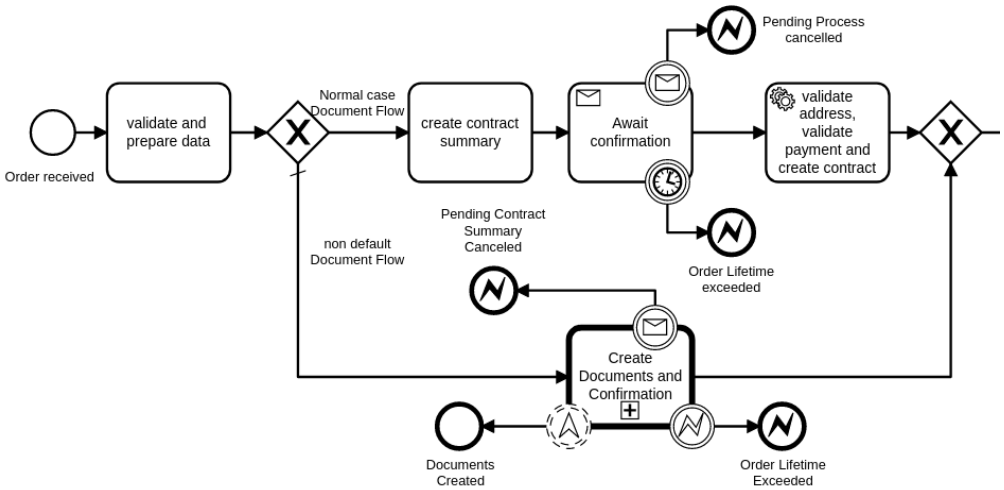


Figure 5.2: Segment of the BPMN model in figure 5.1

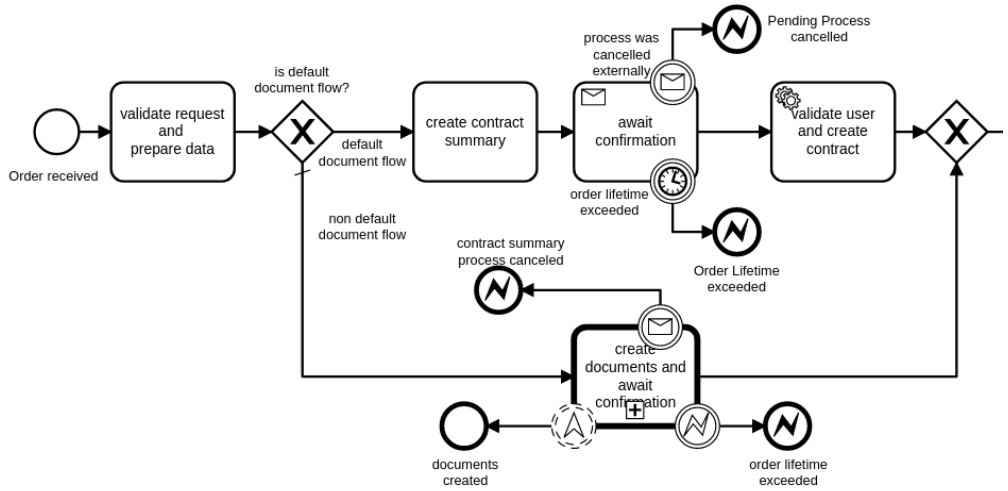


Figure 5.3: Segment 5.2 with changed element labels in order to satisfy naming conventions

The full renaming change can be found in the appendix chapter 6 in figure 1. The full XML representation of this process can be found in the GitHub repository[**?**].

5.3.2 Eliminate Manual Tasks

The only manual task in this BPMN model is the *cancel promotion* task. Canceling promotions can only be done manually by a service agent in this fictitious case and can therefore not be automated. In this case, the manual task can be changed into a *user task* and the advantages of a worklist handler can be utilized as described in section 2.3.2. The original segment with the manual task can be seen in figure 5.4. The corresponding change in this section is shown in figure 5.5.

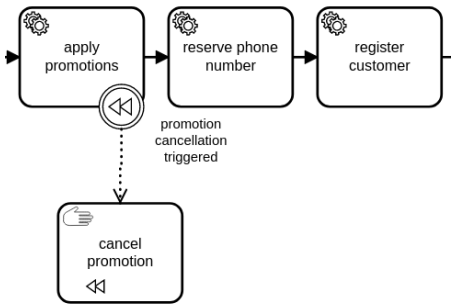


Figure 5.4: Segment of the BPMN model in figure 5.1 with the manual task

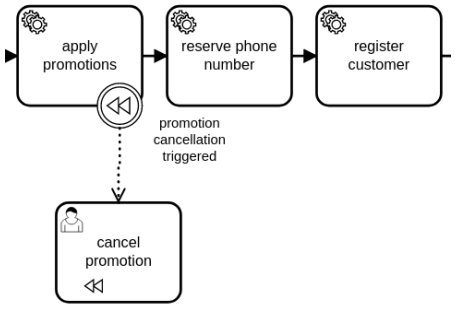


Figure 5.5: Segment that is shown in figure 5.2 with changing the manual tasks to a user task

The fully changed BPMN file can be found in the appendix chapter 6 in figure 2. The full XML representation of this process can be found in the GitHub repository[**?**].

5.3.3 Complete the Process Model

This fictional model covers all cases and error scenarios that can happen in this process.

5.3.4 Extend Automation Boundaries

Since the process model is, apart from the *cancel promotion* task, already fully automated and this activity is not automatable yet the automation boundaries cannot be extended further.

5.3.5 Merge Consecutive Tasks Handled by the Same Resource

According to the rules described in section 4.3.5, the list of elements given in 5.2 can be merged with their direct successor. The segements of the BPMN model with the listed elements can be found in figures 5.6 and 5.8 and the respective change is shown in figures 5.7 and 5.9.

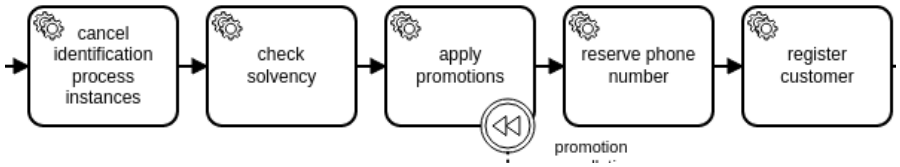


Figure 5.6: Segment of the BPMN model 5.1 with consecutive service tasks

Figure 5.7: Segment that is shown in figure 5.6 with merged consecutive service tasks

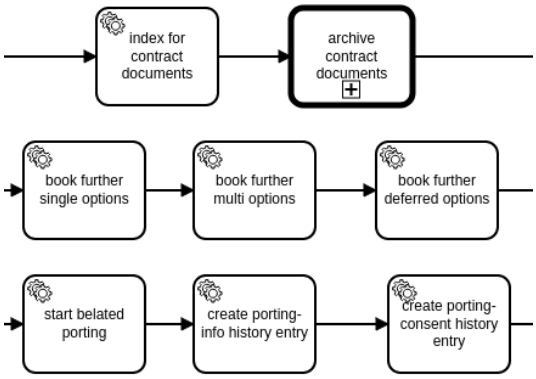


Figure 5.8: Segment of the BPMN model 5.1 with consecutive service tasks

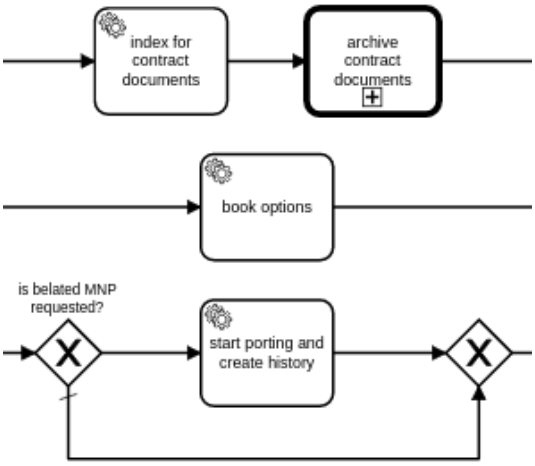


Figure 5.9: Segment that is shown in figure 5.8 with merged consecutive service tasks

The changes in the BPMN file can be found in the appendix chapter 6 in figure 3. The full XML representation of this process can be found in the GitHub repository[**?**].

5.3.6 Replace Combinations of Parallel and Exclusive Gateways With Inclusive Gateways

The only case in this BPMN model where a combination of parallel and exclusive gateways was used instead of an inclusive gateway is in the segment shown in figure 5.10. The corresponding change in this section of the model is shown in figure 5.11.

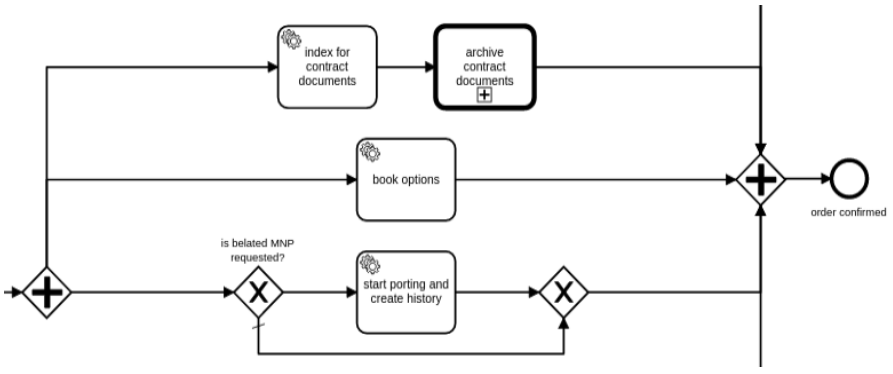


Figure 5.10: Segment of the BPMN model in figure 5.1 where parallel and exclusive tasks are combined instead of using an inclusive gateway

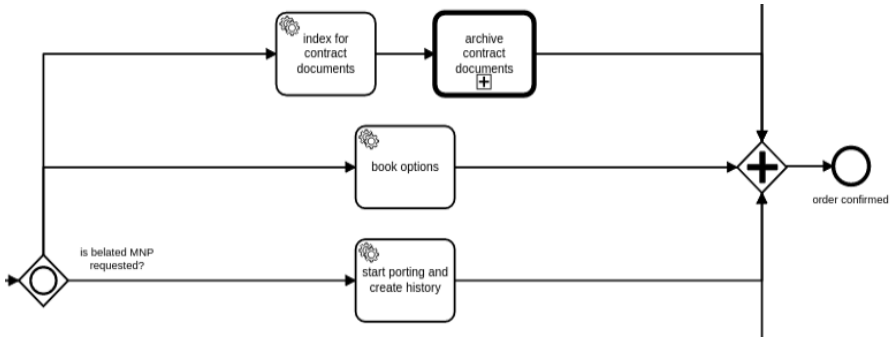


Figure 5.11: Segment that is shown in figure 5.10 with using an inclusive gateway

The changes in the BPMN file can be found in the appendix chapter 6 in figure 4. The full XML representation of this process can be found in the GitHub repository[**?**].

5.3.7 Perform Value Added Analysis and Waste Elimination

To eliminate tasks in a process that have limited value for the customer, a value added analysis as explained in section 3.1.2 will be performed on the customer registration process. For the value added analysis the last version of the process model will be used after implementing the suggestions discussed now.

Following the same approach as in seciton 3.1.2, the process tasks where categorized in RVA (real value adding), BVA (business value adding) and NVA (non value adding). The colored process is shown in the appendix section 6 in figure 5.

For better understanding, the following list will describe the BVA and NVA activities and why they were categorized as NVA/BVA:

Cancel Identification Instances & Check Solvency (BVA)
This activity cancels, in case that a customer registers for the second time, all identification processes of the same customer. Furthermore, the solvency of the customer is checked. Since neither the solvency check, nor the cancellation of identification processes adds value to the customer, but are measures for fraud protection, this activity is marked as business value adding.

Cancel Promotion (NVA)
The canceling promotions activity has only effect when the registration process fails and promotions have already been booked by the customer. In the case that the promotion stays booked and the customer registration fails, the user cannot use the promotion anyway, therefore this activity was categorized as nonvalue adding.

Determine Appropriate Identification Process (BVA)
There are different ways a customer can be identified. This activity decides based on

rules which method is used. Since the whole identification process adds no value to the customer but is used for fraud protection, this activity is marked as business value adding.

Deferred Ident and Contract Confirmation (BVA)
Depending on the decision taken in the previous activity, this activity performs the identification.

Create Index for Contract Documents & Archive Contract Documents (BVA)
Contract documents are not archived for the customer, therefore these two activities are marked as business value adding.

Waste Elimination
Now that the value of each activity was determined, NVA and BVA tasks can be considered for waste elimination.

As mentioned above, the *cancel promotion* task has neither the real customer nor business value and can therefore be eliminated together with the compensation events that were meant to trigger this activity.

Another possible waste elimination could be achieved by evaluating the effect of the implemented fraud protection in the following activities and evaluate if simpler measures for fraud protection can be used to achieve similar results.

- *Cancel identification instances & check solvency (BVA)*,
- *Determine appropriate identification process (BVA)*
- *Deferred ident and contract confirmation (BVA)*

After eliminating the *cancel promotion* activity and everything related to triggering the compensation, the final process model can be found in figure 6.

CHAPTER 6

Conclusion

This thesis aimed to summarize best practices for creating and transforming executable BPMN models. Based on the best practices given in chapter 2 and 3, the following suggestions can be implemented into a given BPMN process model:

1. Comply with naming conventions
2. Eliminate manual tasks
3. Extend automation boundaries
4. Complete the process model
5. Merge consecutive tasks handled by the same resource
6. Replace combinations of parallel and exclusive gateways with inclusive gateways
7. Perform value added analysis and waste elimination

The software described in chapter 4 can be used to scan an *.bpmn* file for violations of the mentioned best practices. The software then returns a list of BPMN elements that violate the given rule. This software is freely available as an open source project on GitHub: `https://github.com/dsunaric/epms-service`.

This thesis goes then on to demonstrate in a case study how the software can be used as an aid to implement the listed suggestions in chapter 5.

While this thesis mentioned a few anti-patterns that are commonly used in executable BPMNS, there are probably many others left that could easily be automatically identified. Further research is needed to quantifiably determine the effectiveness of implementing the given suggestions with measures like the flow-node count or execution time.

List of Figures

Listings

Appendix

BPMN example output

The following output shows the output of the implemented software when giving the example 5.1 BPMN as input.


```
[
{
        "title": "No two consecutive Tasks should be executed by the
           same resource",
        "description": "Two task that are executed one after the
           other can be merged if they have the same enitity
           executing the Task. In case of User tasks this means the
           same usergroup is executing this task. Automated Tasks
           that follow each other should also always be merged to
           minimize flownodes.",
        "details": "The Returned Effected Elements represent a list
           of 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 mergedwith
           its successor on its own. Therefore if \"task1\" , \"task2
           \" and \"task3\" can be merged all together, this
           algorithm will \"task2\" and \"task3\" as effected
           elements",
        "effectedElements": [
        {
                "id": "CancelRelatedWebIdent",
                "name": "Cancel Related Identification Process
                   Instances",
                "type": "Task"
        },
        {
                "id": "ReserveMsisdn",
                "name": "Reserve phone number",
                "type": "Task"
        },
        {
                "id": "BookAutoTopups",
```

```
                "name": "Book further multi options",
                "type": "Task"
        },
        {
                "id": "ServiceTask_0p0tajd",
                "name": "Start Belated Porting",
                "type": "Task"
        },
        {
                "id": "ServiceTask_0e2r82f",
                "name": "Create Porting-Info History Entry",
                "type": "Task"
        },
        {
                "id": "BookSingleTopups",
                "name": "Book further single options",
                "type": "Task"
        }
        ]
},
{
        "title": "No manual Tasks are allowed",
        "description": "Manual Tasks should not be part of an
           executable BPMN diagram",
        "details": "The Returned affected Elements represent a list
           of manual tasks in the given diagram",
        "affectedElements": [
        {
                "id": "data_to_warehouse",
                "name": "store data into datastore",
                "type": "Task"
        }
        ]
},
{
        "title": "Event based gateways instead of a parallel gateway
           followed by exclusive gateways",
        "description": "Event based gateways should be used instead
           of a parallel gateway followed by one or more exclusive
           gateways",
        "details": "This Rule applied whenever a parallel gateways is
            followed by one or more exclusive gateways. The returned
           affected Elements are a list of parallel gateways that are
            followed by one or more exclusive gateways in the given
           diagram",
        "affectedElements": [
        {
                "id": "ParallelGateway_0crudor",
                "name": null,
```

```
59                  "type": "Gateway"
60          }
61          ]
62  },
63  {
64          "title": "Apply naming conventions",
65          "description": "One aspect of naming convention is the length
                of labels, this rule scanns the bpmn for rlements with
               long labels (> 5 words)",
66          "details": "The returned affected elements are all elements
               with a label that has more than 5 words",
67          "affectedElements": [
68          {
69                  "id": "
                       ServiceTask_DefaultDocWorkflowPostAwaitConfirmation
                       ",
70                  "name": "validate address, validate payment and
                       create contract",
71                  "type": "Task"
72          },
73          {
74                  "id": "DecideIdAndCsProcess",
75                  "name": "Decide non default doc flow and
                       idnetification Process",
76                  "type": "Task"
77          },
78          {
79                  "id": "save_member_for_riskident",
80                  "name": "save new member in backend service",
81                  "type": "Task"
82          }
83          ]
84  }
85  ]
```

Case Study - Implemented Aprovements

The following section shows the improvement steps done in section 5.3.

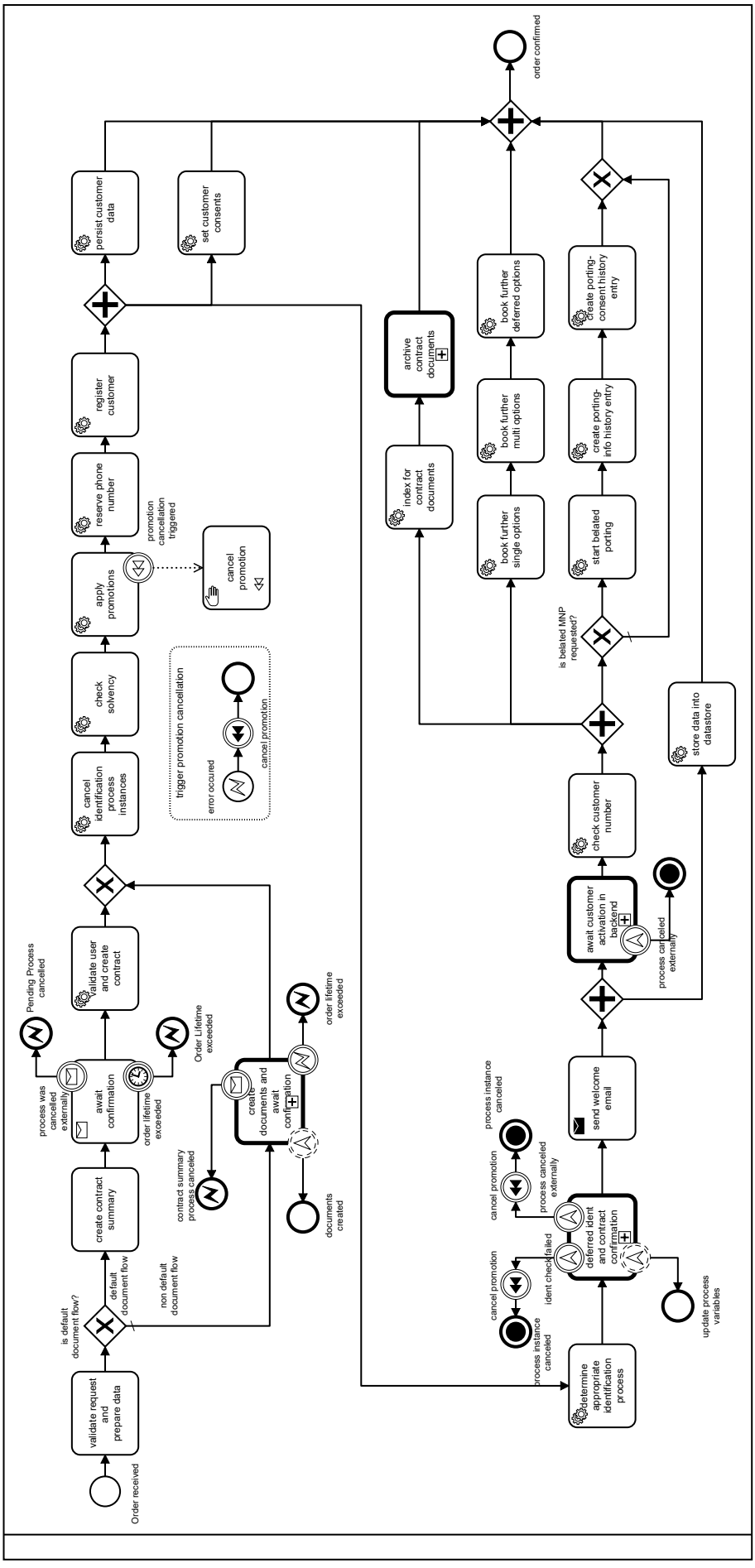


Figure 1: Adapted customer registrations process, after renaming

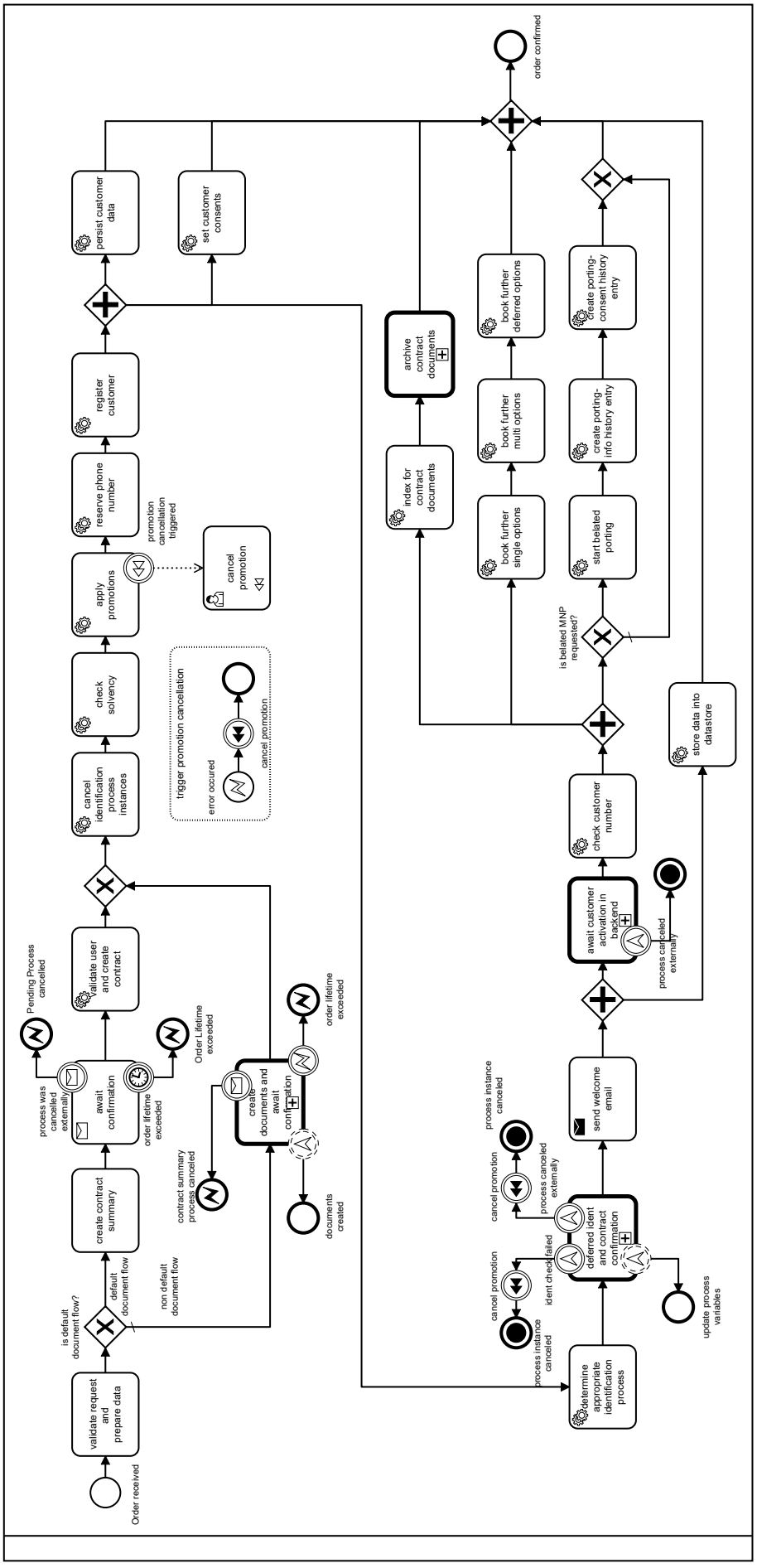


Figure 2: Adapted customer registrations process, after changing the manual task to a user task

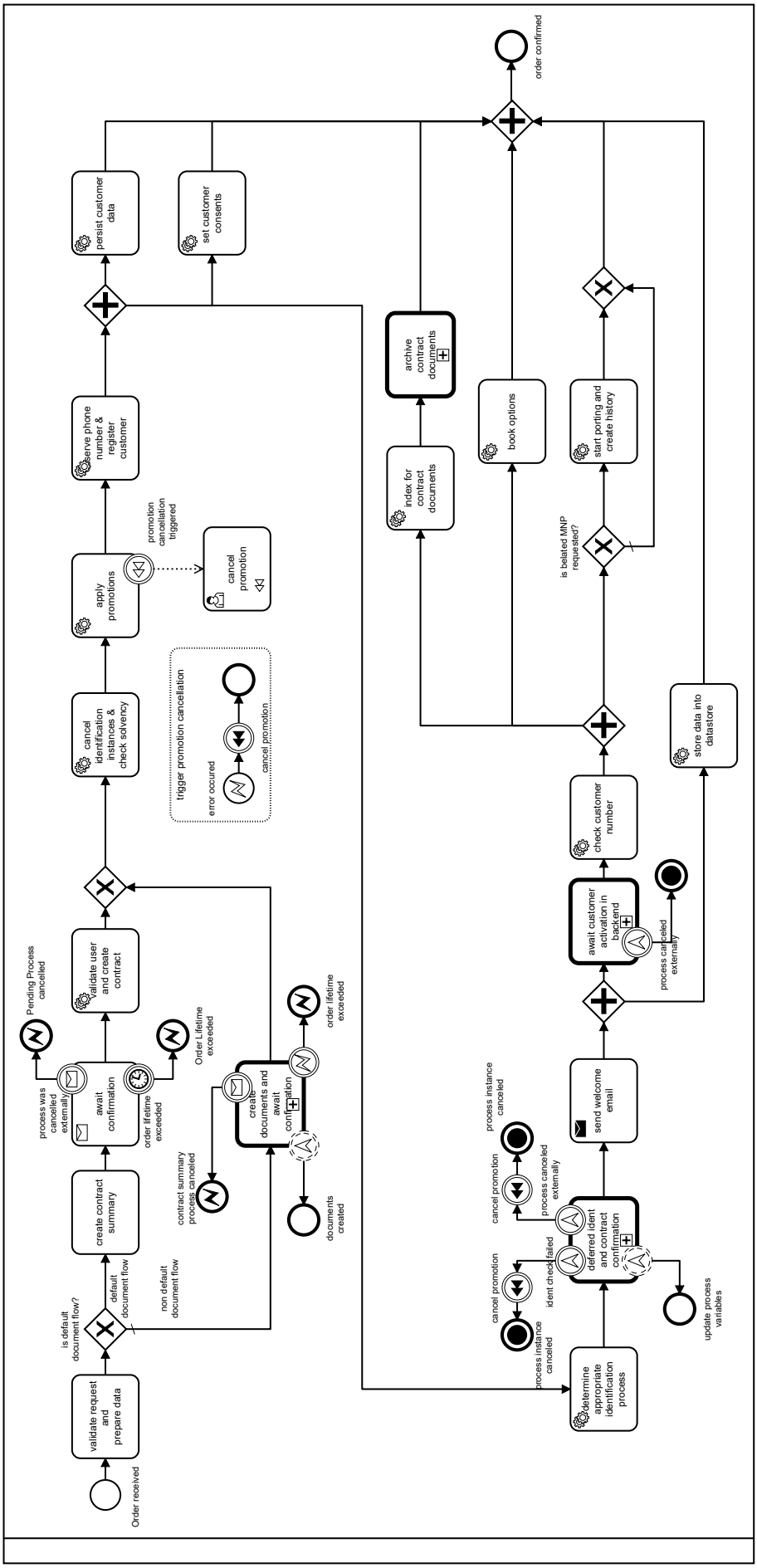


Figure 3: Adapted customer registrations process, after merging tasks that are executed by the same resource

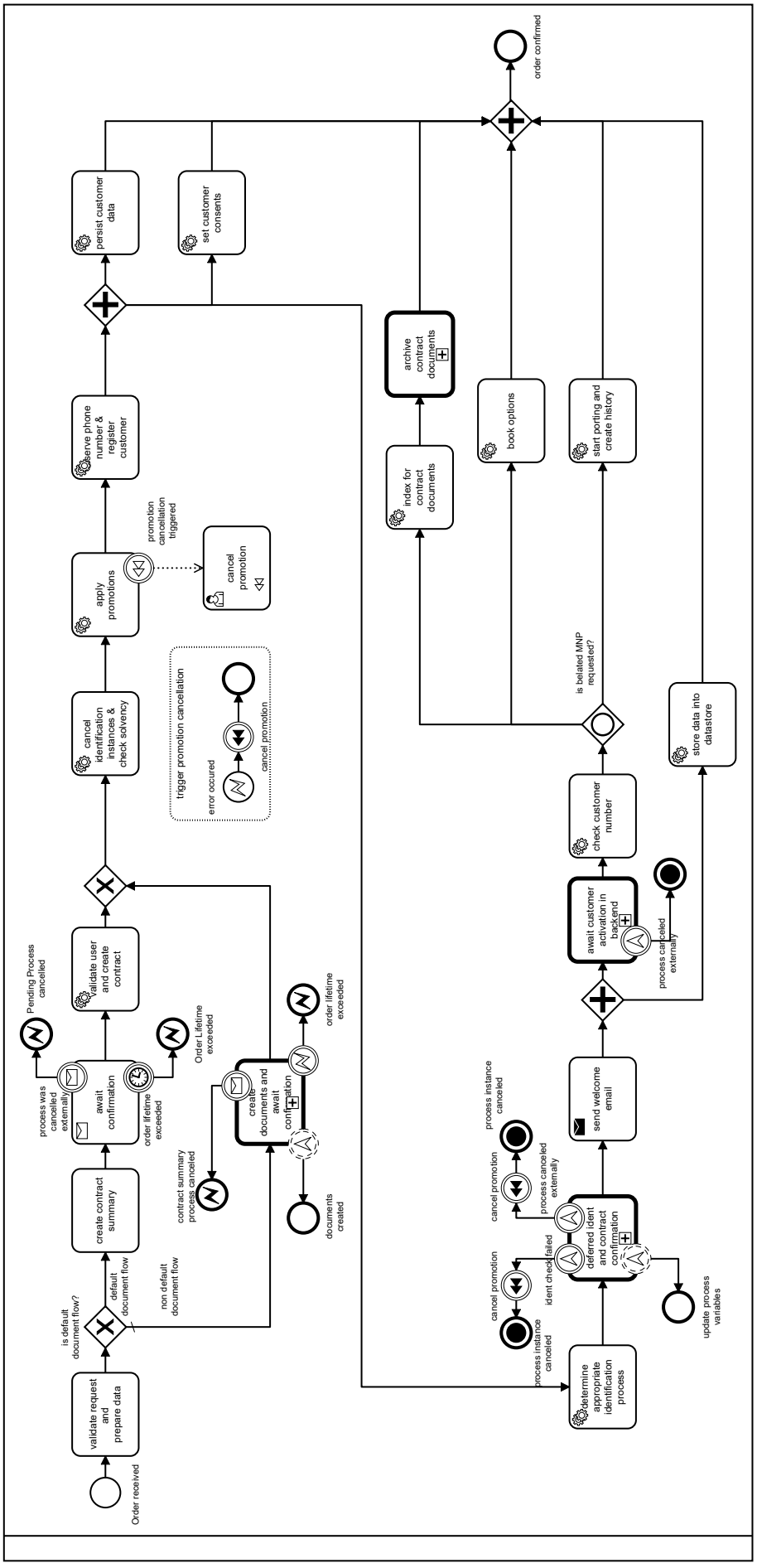


Figure 4: Adapted customer registrations process, after changing the combined parallel and exclusive gateways to an inclusive gateway

Case Study - Value Added Analysis

The following section shows resources needed for the implementation of value added analysis in section 5.3.7.

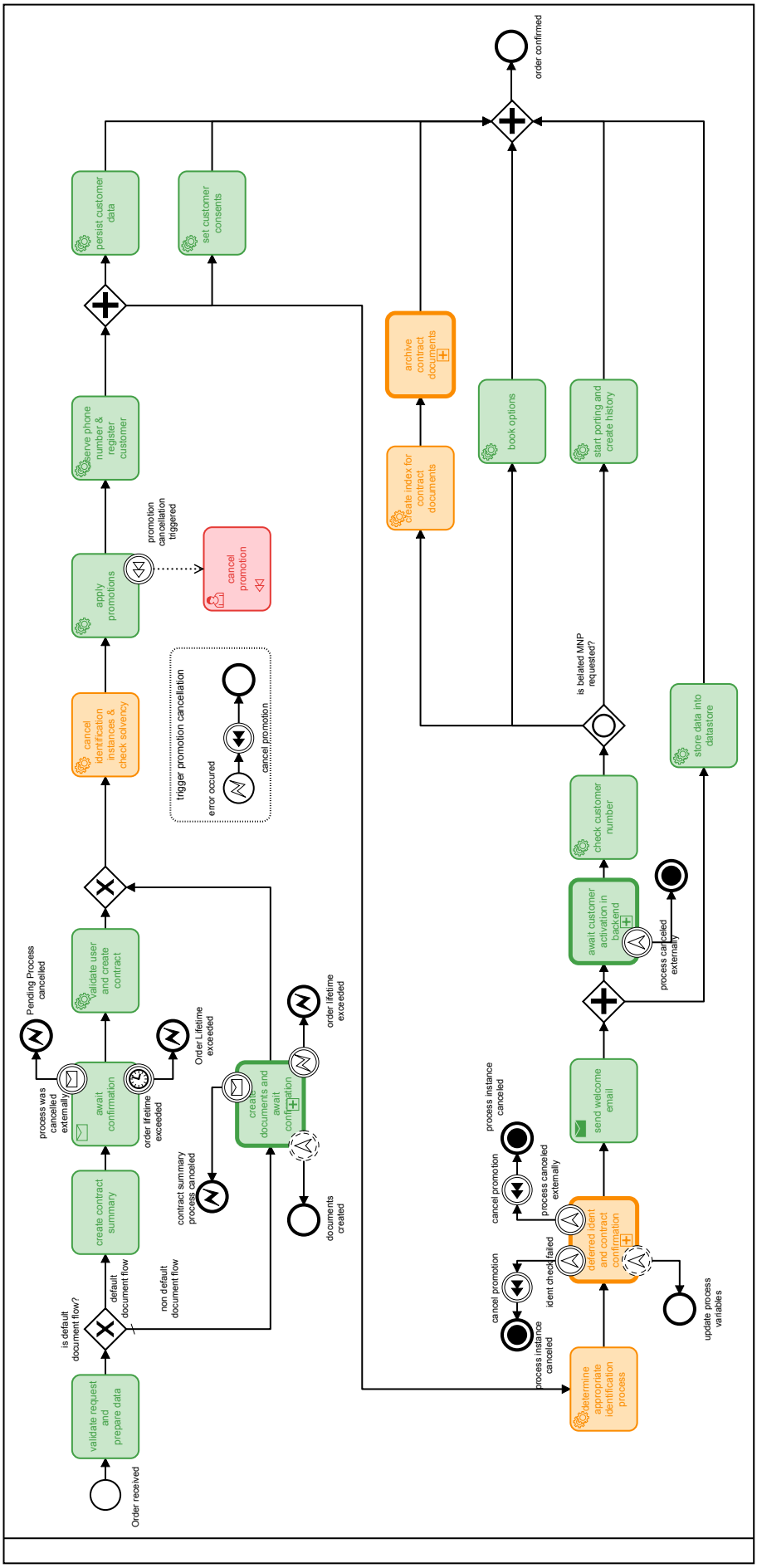


Figure 5: Adapted customer registrations process with colored in activities according after value added analysis

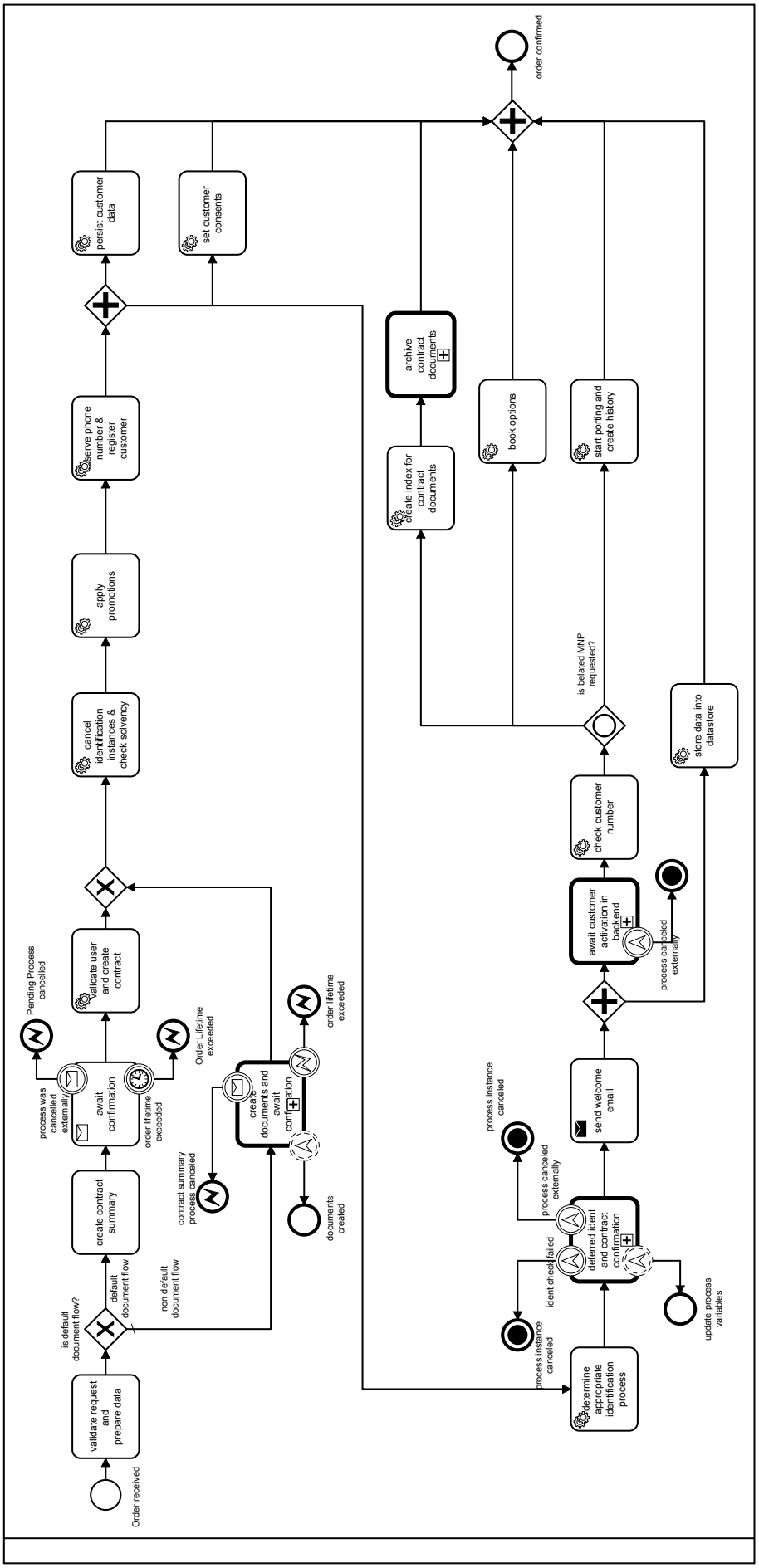


Figure 6: Adapted customer registrations process after waste elimination