

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL  
INSTITUTO DE INFORMÁTICA  
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LEONARDO SILVA ROSA

**A visual approach for identification and  
annotation of business process elements in  
process descriptions**

Work presented in partial fulfillment  
of the requirements for the degree of  
Bachelor in Computer Science

Advisor: Prof. Dr. Lucineia Heloisa Thom

Porto Alegre  
November 2020

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof. Carlos André Bulhões Mendes

Vice-Reitora: Prof<sup>ª</sup>. Patricia Helena Lucas Pranke

Pró-Reitora de Graduação: Prof<sup>ª</sup>. Cíntia Inês Boll

Diretora do Instituto de Informática: Prof<sup>ª</sup>. Carla Maria Dal Sasso Freitas

Coordenador do Curso de Ciência de Computação: Prof. Sérgio Luis Cechin

Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

## ABSTRACT

Business Process Management (BPM) has been proven to provide several benefits for organizations (e.g., efficiency, agility, governance). However, the effort required for adopting a process-centered approach can be a challenge in different aspects, including financial concerns, organizational changes and time consumption. In particular, process modeling is a complex, but crucial activity constantly performed during the process discovery and redesign of the BPM life-cycle. Most companies already have their processes described as text documents (process descriptions), which can be leveraged by business analysts as one of the techniques to effectively build process models. In light of this, this study proposes an approach to enhance process descriptions as complementary artifacts in which BPMN 2.0 core process elements are identified and interactively annotated with visualization features. Specifically, the presented approach is able to detect sequences of words that indicate the presence of a process element and create a metadata structure which is exposed through a consumable web service and further used for generating the annotated process description. As means to demonstrate the approach applied in practice, a prototype was developed, which is capable of annotating the identified process elements by adding visual features such as colors and BPMN 2.0 symbols. In order to evaluate the approach, two separate experiments were conducted, performing a user focused survey and a business process model design case study, respectively. The first experiment shows promising results in every category evaluated, specially concerning the usefulness of the approach to assist the process modeling phase, for which 88% of the users indicated positive results. The second experiment shows a process description with a precision of 77% of annotated process elements in comparison to its original process model. Additionally, several possible use cases enabled by this study are presented as well as ideas for complementary future researches.

**Keywords:** Business process management. Natural language processing. Visually interactive process description.

## **Uma abordagem visual para identificação e anotação de elementos de processos de negócio em descrições de processos**

### **RESUMO**

Gerenciamento de Processos de Negócio (BPM) tem se provado capaz de oferecer diversos benefícios para organizações (e.g., eficiência, agilidade, governança). Entretanto, o esforço necessário para adotar uma abordagem centralizada em processos pode ser desafiador em diversos aspectos, incluindo questões financeiras, modificações organizacionais e consumo de tempo. Particularmente, a modelagem de processos é uma tarefa complexa, mas crucial, executada constantemente durante as fases de descoberta e redesenho de processos no ciclo de vida de BPM. Grande parte das empresas já possui seus processos descritos em documentos de texto (descrições de processo), o que pode ser aproveitado por analistas de negócio como uma das técnicas para de forma efetiva construir modelos de processo. Diante deste contexto, este estudo propõe uma abordagem para incrementar descrições de processos, gerando artefatos auxiliares nos quais principais elementos da BPMN 2.0 são identificados e interativamente anotados com características visuais. Especificamente, a abordagem apresentada é capaz detectar sequências de palavras que indicam a presença de um elemento de processo e criar uma estrutura de metadados que é exposta através de um serviço da web a ser consumido e posteriormente usada para gerar a descrição de processo anotada. Como forma de demonstrar a abordagem aplicada na prática, um protótipo foi desenvolvido, sendo capaz de anotar os elementos identificados a partir da utilização de elementos visuais como cores e símbolos da BPMN 2.0. Para avaliar a abordagem, dois experimentos separados foram conduzidos, realizando uma pesquisa focada em usuários e um estudo de caso para construção de modelos de processo de negócio, respectivamente. O primeiro experimento mostrou resultados promissores em todas as categorias avaliadas, especialmente se tratando da utilidade da abordagem para auxiliar a fase de modelagem de processo, para a qual 88% dos usuários indicaram resultados positivos. O segundo experimento mostra uma descrição de processo com uma precisão de 77% de elementos de processo anotados em comparação com o seu modelo original. Além disso, diversos casos de uso possibilitados por este estudo são apresentados, assim como ideias para futuras pesquisas complementares.

**Palavras-chave:** Gerenciamento de processos de negócio. Processamento de linguagem natural. Descrições de processo visualmente interativas.

## LIST OF FIGURES

Figure 2.1	Subset of BPMN 2.0 elements.....	16
Figure 2.2	Restaurant Service process model according to the BPMN 2.0 .....	16
Figure 2.3	NLP pipeline.....	18
Figure 2.4	Example of constituency parsing in CoreNLP .....	19
Figure 4.1	Approach to identify and annotate process elements in process descriptions	32
Figure 5.1	Approach phases in a client-server model.....	48
Figure 5.2	Class diagram of the server component.....	52
Figure 5.3	Sequence diagram of the server component .....	54
Figure 5.4	Class diagram of the client component.....	55
Figure 5.5	Sequence diagram of the client component .....	56
Figure 5.6	Input text.....	57
Figure 5.7	Output Screen .....	58
Figure 5.8	Customize color.....	58
Figure 5.9	Annotated text.....	59
Figure 6.1	First section of the questionnaire.....	61
Figure 6.2	First sentence in the second section of the questionnaire .....	62
Figure 6.3	Third section of the questionnaire .....	63
Figure 6.4	Fourth section of the questionnaire.....	65
Figure 6.5	Annotation comprehension.....	68
Figure 6.6	Annotation comprehension per occupation .....	68
Figure 6.7	Annotation correctness .....	70
Figure 6.8	Annotation correctness per BPMN experience.....	70
Figure 6.9	Annotation usefulness for model design .....	73
Figure 6.10	Annotation usefulness for model design per modeling experience .....	73
Figure 6.11	Actors annotation comparison .....	75
Figure 6.12	Tasks annotation correlation .....	76
Figure 6.13	Exclusive decisions annotation comparison .....	76
Figure 6.14	Parallel executions annotation comparison.....	77
Figure 7.1	Computer Repair annotated .....	80
Figure 7.2	Hotel Service annotated.....	81
Figure 7.3	Computer Repair (original) annotated .....	83
Figure 7.4	Results of identified elements in the original text .....	84
Figure 7.5	Computer Repair process model.....	86
Figure 7.6	Hotel Service process model .....	86
Figure 7.7	Computer Repair original process model .....	87
Figure 7.8	Hotel Service original process model .....	87
Figure A.1	Participant annotation 1 .....	95
Figure A.2	Participant annotation 2 .....	95
Figure A.3	Participant annotation 3 .....	96
Figure A.4	Participant annotation 4 .....	96
Figure A.5	Participant annotation 5 .....	97
Figure A.6	Participant annotation 6 .....	97
Figure A.7	Participant annotation 7 .....	97
Figure A.8	Participant annotation 8 .....	97

## LIST OF TABLES

Table 2.1	Main properties of the action data structure .....	21
Table 3.1	Automated extraction of business process information.....	24
Table 3.2	Generation of business process descriptions .....	27
Table 3.3	Comparison of related works with the approach to be presented in this study	31
Table 4.1	Adaptation of a process description .....	34
Table 5.1	Classes in relation to phases of the presented approach.....	53
Table 7.1	Variable values for precision and recall .....	82

## LIST OF ALGORITHMS

1	Algorithm for detecting possible events .....	37
2	Algorithm for generating the JSON metadata .....	39
3	Algorithm for filtering and classifying process elements .....	39
4	Algorithm for identifying the start index of elements .....	40
5	Algorithm for identifying the end index of elements.....	41
6	Algorithm for identifying the indices of a split gateway branch .....	42
7	Algorithm for identifying the indices of join gateways .....	43
8	Algorithm for merging properties of the JSON metadata.....	44
9	Algorithm for assigning each word to its process element type .....	45
10	Algorithm for adding visual features in a text structure .....	46

## **LIST OF ABBREVIATIONS AND ACRONYMS**

BPM	Business Process Management
BPMN	Business Process Model and Notation
NLP	Natural Language Processing
POS	Part-of-speech
PME	Process Model Extraction
HTTP	Hypertext Transfer Protocol
DOM	Domain Object Model
SPA	Single Page Application
API	Application Programming Interface
JSON	JavaScript Object Notation
SoC	Separation of Concerns
CSS	Cascading Style Sheets
CBSE	Component-based Software Engineering



## CONTENTS

<b>1 INTRODUCTION</b> .....	<b>10</b>
<b>2 BACKGROUND</b> .....	<b>14</b>
<b>2.1 Business process concepts</b> .....	<b>14</b>
<b>2.2 Natural language processing</b> .....	<b>17</b>
2.2.1 NLP pipeline .....	17
2.2.2 Stanford CoreNLP toolkit .....	19
<b>2.3 Extraction of business process elements</b> .....	<b>20</b>
<b>2.4 Human-computer interaction</b> .....	<b>22</b>
<b>3 RELATED WORKS</b> .....	<b>24</b>
<b>3.1 Automated extraction of business process information</b> .....	<b>24</b>
<b>3.2 Generation of business process descriptions</b> .....	<b>27</b>
<b>4 IDENTIFICATION AND ANNOTATION OF BUSINESS PROCESS ELE- MENTS</b> .....	<b>32</b>
<b>4.1 Textual guidelines</b> .....	<b>32</b>
<b>4.2 Input handling</b> .....	<b>35</b>
<b>4.3 Elements extraction</b> .....	<b>35</b>
<b>4.4 Text identification</b> .....	<b>38</b>
<b>4.5 Text annotation</b> .....	<b>44</b>
<b>5 PROTOTYPE</b> .....	<b>47</b>
<b>5.1 Architecture</b> .....	<b>47</b>
<b>5.2 Technologies</b> .....	<b>48</b>
<b>5.3 Server component</b> .....	<b>50</b>
<b>5.4 Client component</b> .....	<b>54</b>
<b>5.5 Demonstration</b> .....	<b>57</b>
<b>6 EVALUATION OF THE APPROACH</b> .....	<b>60</b>
<b>6.1 Questionnaire structure</b> .....	<b>60</b>
<b>6.2 Result analysis</b> .....	<b>66</b>
6.2.1 Comprehension of the annotated description.....	67
6.2.2 Annotation correctness.....	69
6.2.3 Usefulness for modeling purposes .....	72
6.2.4 Effectiveness of the approach .....	74
<b>6.3 Discussion</b> .....	<b>78</b>
<b>7 CASE STUDY</b> .....	<b>79</b>
<b>8 CONCLUSION</b> .....	<b>88</b>
<b>REFERENCES</b> .....	<b>91</b>
<b>APPENDIX A — ANNOTATIONS SUBMITTED IN THE QUESTIONNAIRE</b> ...	<b>95</b>

## 1 INTRODUCTION

To interact within itself or with customers and business partners, every organization, profitable or not, has to manage a number of processes. A business process is a collection of inter-related events, activities and decision points that involve a number of actors and objects, and that collectively lead to an outcome that is of value to at least one customer (DUMAS et al., 2018). In order to be competitive, companies aim to improve the efficiency and quality of their business processes and adopt well established standards of specifications and execution (THOM; REICHERT; IOCHPE, 2009). Thus, *Business Process Management* (BPM) has increasingly being adopted by organizations due to its capabilities of improving operational performance, enhancing service quality, ensuring regulations and compliance as well as reducing costs (DUMAS et al., 2018). BPM encompasses the entire lifecycle of a business process, which means a continuous cycle composed by several phases that are addressed throughout the existence of the process.

According to Dumas et al. (2018), the starting point of the BPM lifecycle is the *Process identification*, with the purpose of delimiting which processes address certain business problems. The *Process discovery* phase follows in order to document and detail the identified processes. Successively, analysing the discovered processes based on a set of predefined metrics and measures is the key phase for spotting bottlenecks and potential points of improvement. The redesign and implementation are the subsequent phases in which process changes addressing the identified issues are proposed and applied, respectively. Finally, monitoring and controlling the redesigned process execution aims to detect possible new issues or deviations to the intended behaviour, thus, restarting the cycle if necessary. Although BPM encompasses the entire lifecycle of a business process, this study focus on the *Process Discovery* phase.

There are different representation forms and notations with which an organization can document its business processes (ZIMOCH et al., 2018). Process descriptions are a common form which intends to describe a business process in the form of textual representation. Researches have shown that 85% of the companies' information are represented by non-structured formats, such as text documents (BLUMBERG; ATRE, 2003). As textual descriptions may not be sufficient to represent every aspect of a process (ALLWEYER, 2009), organizations also adopt *business process models* as a representation of their processes (KETTINGER; TENG; GUHA, 1997), which comprises visual elements and syntactic rules. Further, current standardization efforts points to the *Business Process*

*Model and Notation* (BPMN) (OTTENSOOSER et al., 2012), in particular, BPMN 2.0 (COMMISSION et al., 2013), as the standard notation to follow.

Before creating a model, it must be discovered and understood how the process is performed. During the Discovery phase of the BPM lifecycle, the process analyst may use several methods to achieve this goal, such as: *Evidence-Based*, *Interview-Based* and *Workshop-Based* methods (DUMAS et al., 2018). The *Interview-Based* discovery uses techniques for the purpose of extracting information about the process execution by interviewing the domain experts. Furthermore, the *Workshop-Based* discovery benefits from the interaction and contribution of several participants belonging to different roles, such as domain experts, process owner, process analyst. Finally, in the *Evidence-Based* discovery, the analyst studies how a process works, using methods such as Document Analysis, Observation and Automatic Process Discovery.

With regard to the latter, new techniques have progressively been proposed in recent researches. Process Mining, for example, aims to discover, monitor and improve processes extracting knowledge from event logs available in information systems (AALST, 2012). Such logs, however, are not always present in organizations with a simpler technological infrastructure. This study, on the other hand, proposes an approach based on a Natural Language Processing (NLP) technique (LIDDY, 2001) applied to existing documented process descriptions. Nevertheless, regardless of which methods and techniques are used, many challenges may be presented, for instance, different interpretations of the process by the employees, ambiguous information in text documents, unclear or misleading material. Therefore, *process modeling* can become a complex, time consuming and error prone activity (DUMAS et al., 2018).

In light of these challenges, many studies have shown promising results in automating or helping the modeling of processes (FRIEDRICH; MENDLING; PUHLMANN, 2011; HONKISZ; KLUZA; WIŚNIEWSKI, 2018; FERREIRA et al., 2017; SILVA et al., 2019). In particular, the research of Silva et al. (2019) establishes a service-oriented architecture for enabling sound process description generation from a process description written in natural language and it is able to mark and annotate business process elements (e.g., Tasks, Start and End Events, Gateways) in the generated process description. In the context of this study, a complementary re-reading of Silva's research is presented in this study, focusing on *a user-interactive visual approach for identification and annotation of business process elements in process descriptions*. As it stands, the text annotation in Silva's study is not applicable to the given original process description, but rather re-

stricted to the generated sound process description. Furthermore, whilst Silva's approach sets a starting point for visualization features in process descriptions, it is limited to modifying the font color of words in the text with a preselected set of colors. The approach proposed in this study, on the other hand, aims to provide a user-interactive text annotated by a variety of visual features (e.g., symbols, colors, highlighters) and based on the original process description.

The main goal of the approach presented in this study is to provide a valuable annotated process description based on flexible and customized visual features which intends to be beneficial as an artefact for organizations and applicable for several use cases. A study performed by Ottensooser et al. (2012) conducted an experiment in order to compare the efficiency between textual, graphical and both notations for the understanding of a process. The analysis of the experiment results suggests that all groups of participants benefited from leveraging both notations. According to the authors, while a pure textual notation is limited to a linear order, the pure graphical notation causes a cognitive overload. A previous study of Sweller, Merriënboer and Paas (1998) observed that, when reading diagrams, the human brain memory is restricted to about 30 seconds of storage duration and roughly seven matters.

The approach presented in this study basis itself on the fact that the textual notation (i.e., process descriptions) is helpful for the comprehension of the process and hypothesizes that a mixed notation based on the process description annotated by visual features may assists even further on such task. Additionally, a use case is demonstrated on how the annotated process description can help to build a process model. The use cases, however, are not limited to building process models and comprehending the process. An annotated process description may also be useful for teaching purposes, helping to track changes in the process and others.

An additional key motivation for this study is due to the lack of functional implementations of related approaches in this field. In a study performed by Riefer, Ternis and Thaler (2016), several approaches for mining process models from natural language texts are analyzed and compared. According to the authors, however, this comparison was restricted to a theoretical basis, due to the absence of available implementations. The approach proposed in this study is complemented by a prototype which enables a number of use cases for end-users. Furthermore, this prototype was designed with the goals of being reproducible and extended by future researches.

This study is organized as follows. Chapter 2 presents key definitions and con-

cepts related to the approach proposed in this study. Chapter 3 introduces and discusses related works regarding the process discovery phase. Chapter 4 presents the proposed approach and key algorithms adopted in this study. Chapter 5, shows the presented approach in practice, detailing the development of the tool. The evaluation of the approach is performed and analyzed in Chapter 6, in which a first experiment is conducted. Following, the second experiment in Chapter 7 demonstrates an use case of the approach in the context of creating a process model. Finally, Chapter 8 concludes this study, discussing positive and improvement points as well as presenting ideas for possible future complementary researches.

## 2 BACKGROUND

This chapter presents the key theoretical background referenced in further chapters of this study. First, it defines basic business process concepts required for the construction of a *process description*, introduced in Chapter 1. Second, it introduces NLP concepts, the NLP pipeline and the toolkit used in the context of this study. Finally, this chapter presents a state of the art study used as basis for the extraction of process elements phase of this study.

### 2.1 Business process concepts

A business process encompasses a number of activities, events and decision points. Furthermore, it involves a number of actors (i.e., human actors, organizations or software systems) (DUMAS et al., 2018). Understanding the definitions of those terms is crucial for representing a business process, regardless of the form or notation used. To exemplify these definitions, a hypothetical business process of a restaurant interacting with a customer is introduced below in the form of a process description.

#### Restaurante Service

1 *When a customer arrives at the restaurant, the host verifies the number of tables avail-*  
 2 *able. If one or more tables are available, the host calls the waiter to guide the customer*  
 3 *to one table. Otherwise, the host informs the customer to wait. Once a table has been*  
 4 *cleared, the waiter guides the customer and gives the menu. After five minutes, the*  
 5 *waiter returns and asks for the order. The customer may order food and/or drinks.*  
 6 *While the kitchen prepares the order, the waiter prepares the table. As soon as the*  
 7 *order is ready, the waiter brings it to the table. When the customer finishes his meal,*  
 8 *he pays for the order and leaves the restaurant.*

As part of a subset of activities, tasks are considered actions that have a time duration and represent units of work. They are generally performed by human actors involved in the business process, defined as process participants (DUMAS et al., 2018). In a study performed by Ferreira et al. (2017), in a process description, tasks are mostly described by verbal sentences in the present or in the future tense. As such, sentences representing tasks are usually comprised by verbs, objects and subjects. In the presented

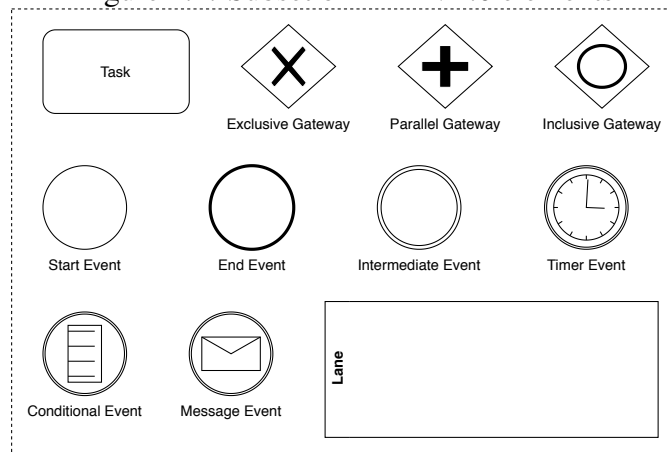
process description, examples of tasks are: “verifies the number of tables available” (lines 1-2), “prepares the order” (line 6) and “pays for the order” (line 8).

Events correspond to things that happen atomically, meaning that they have no duration. They are generally used to specify the occurrence of a certain action that may trigger the execution of a series of following activities (DUMAS et al., 2018). Usually, they are represented in sentences by using verbs in the past or present perfect tense, for example: “Once a table has been cleared” (lines 3-4). In some cases, however, the verb may also appear in other verbal tenses: “When a customer arrives at the restaurant” (line 1). Additionally, events may also represent the occurrence of a specific temporal event (DUMAS et al., 2018), for example: “After five minutes” (line 4). Though there are a number of further usages for events, such details go beyond the scope of this study.

Decision points are points in time when a decision is made such that the execution flow of the business process is affected (DUMAS et al., 2018). For the context of this study, three types of decision points are introduced. *Exclusive decisions* model the relation between two or more alternative activities, meaning that only one resulting alternative must be executed. For example: “If one or more tables are available (...).Otherwise (...)” (lines 2-3). *Parallel executions* occur when two or more activities do not have any order dependencies on each other, meaning that one does not require or exclude the other, thus, being able to be executed concurrently. For example: “While the kitchen prepares the order, the waiter prepares the table” (line 6). Finally, *inclusive decisions* model situations when the result of a decision may lead to one or more outcomes executed, i.e., multiple cases can be true at the same time, but not necessarily all of them. For example: “The customer may order food and/or drinks” (line 5).

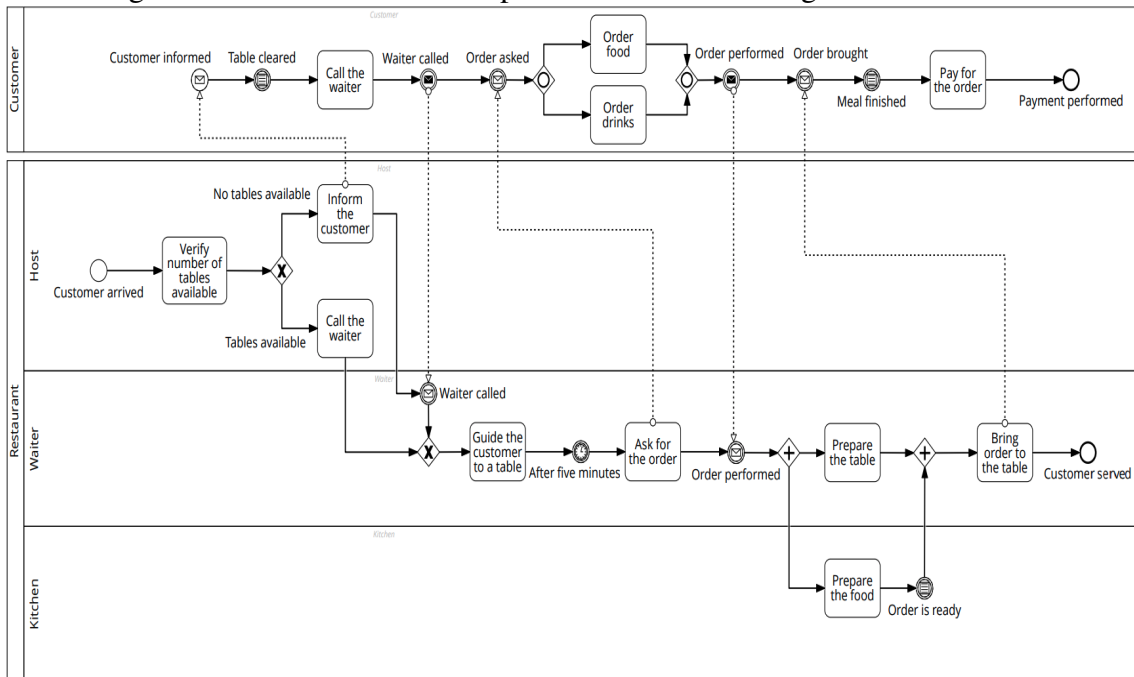
According to the modeling notation followed in this study, the BPMN 2.0 (COMMISSION et al., 2013), an *activity* is a work that is performed within a business process. It can be categorized in three different types: task, sub-process and call activity. An *event*, on the other hand, is something that happens during the course of the process, hence, it has a cause and usually a consequence, which requires or enables a certain reaction. Furthermore, in the BPMN 2.0, a decision point is represented as a *gateway* and is used to control the flow of execution of the process. The execution flow may either diverge, creating branches of different flows or converge to a same flow of execution. Finally, the actors of the process are represented as *lanes* which are often used to delimit a sub-partition of the process performed by the respective actor. Figure 2.1 shows the subset of considered BPMN 2.0 elements in the approach proposed in this study. Additionally, Figure 2.2

Figure 2.1: Subset of BPMN 2.0 elements



Source: The Authors

Figure 2.2: Restaurant Service process model according to the BPMN 2.0



Source: The Authors

shows the process description introduced in this section in the form of a process model according to BPMN 2.0.

Although process descriptions can be a simple form of representing a business process, there are issues which may be present, jeopardizing the comprehension of the process. One common difficulty faced by business analysts when analyzing process descriptions is the presence of ambiguity (SILVA et al., 2018). It can be manifested in several forms, such as the usage of pronouns (e.g., “he”, “she”, “this”) which do not clarify the referenced word, or unclear subject and/or objects in the sentence. Nevertheless, the type of ambiguity emphasized for the context of this study is regarding part of speech.



In many cases, words can have multiple classifications, when taken out of context. For example, the following words can be classified as nouns or verbs: “check”, “fix”, “access”, “control”, “cut”. Such cases may disrupt the intended meaning of the sentence and are particularly problematic for NLP based approaches.

## **2.2 Natural language processing**

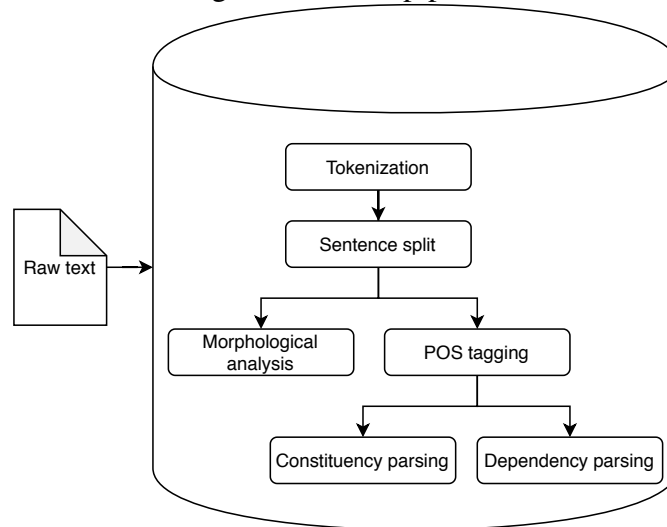
In order to introduce NLP, first, it is necessary to address the concept of a natural language. According to Lyons (1991), a natural language is defined as any kind of human language emerged with no prior planning and which has evolved through its own usage. Every Latin, Greek, Arabic based languages are examples of natural languages, and the variety of existing languages and dialects in the present days is a proof that natural languages are in constant transformation. On the other hand, artificial languages, such as programming languages, are premeditated, constructed and formalized before they are used in practice, thus differentiating themselves from the definition of natural languages.

The aim of a linguistic science is to characterize and explain observations such as the cognitive side of how humans acquire, understand and produce language as well as the linguistic structures used in language communication (MANNING; SCHUTZE, 1999). In the context of NLP, scientists struggle to agree upon a single definition. However, a simplified definition from Liddy (2001) describes NLP as a motivated range of computational techniques for analyzing and representing naturally occurring texts (i.e., natural language), in different levels of linguistic analysis with the goal of achieving a human-level understanding.

### **2.2.1 NLP pipeline**

In the recent years, several studies, technologies and implementations have emerged in the field of NLP. It is a consensus that extracting meaningful information from text resources tends to be decomposed into a number of stages, each of which accomplishes a single goal and is based on the result of the previous stage (INDURKHYA; DAMERAU, 2010). The sequential combination of those stages is defined as the *NLP pipeline* and can be split into three general categories: syntactic analysis, semantic analysis and pragmatic analysis. A deeper level of granularity of each category leads to one or more sequential

Figure 2.3: NLP pipeline



Source: The Authors

stages, those of which can vary between different approaches of NLP. Figure 2.3 shows the stages of the pipeline which are used as ground for the NLP phase of this study.

As the first step of every NLP approach, the tokenization is responsible for splitting the text into a sequence of words (i.e., tokens) and must be able to handle apostrophes, hyphens and further edge cases which may appear in texts. Next in the pipeline, punctuation tokens generated in the previous step are identified in order to split the text into individual sentences. As soon as both stages have been performed, each generated token can be classified in regard to its respective part-of-speech (POS), which indicates the syntactical role in the text. Similarly dependent of the first two steps, the morphological analysis provides the base form of each token, which means identifying plurals and verbal conjugations and retrieve the base words that originate them.

After the POS categorization, a syntactic analysis of each sentence of the text can be performed in order to obtain the relation between the tokens. The sentence is parsed in accordance with a predefined model, which dictates how a token is related to another in a certain situation. A sentence can be syntactically analyzed based on two different parsing methods. The constituency parsing represents the sentence in the form of tree-like nested sub-phrases, defined as *phrase structures* (CHOMSKY; LIGHTFOOT, 2002), in which each non-terminal node contains a phrase level label, while each terminal node contains a word level label. The dependency parsing, on the other hand, provides the *grammatical relation* between each token (MARNEFFE et al., 2006) based on the predefined model. As such, different models can generate different labels and grammatical relations between tokens.

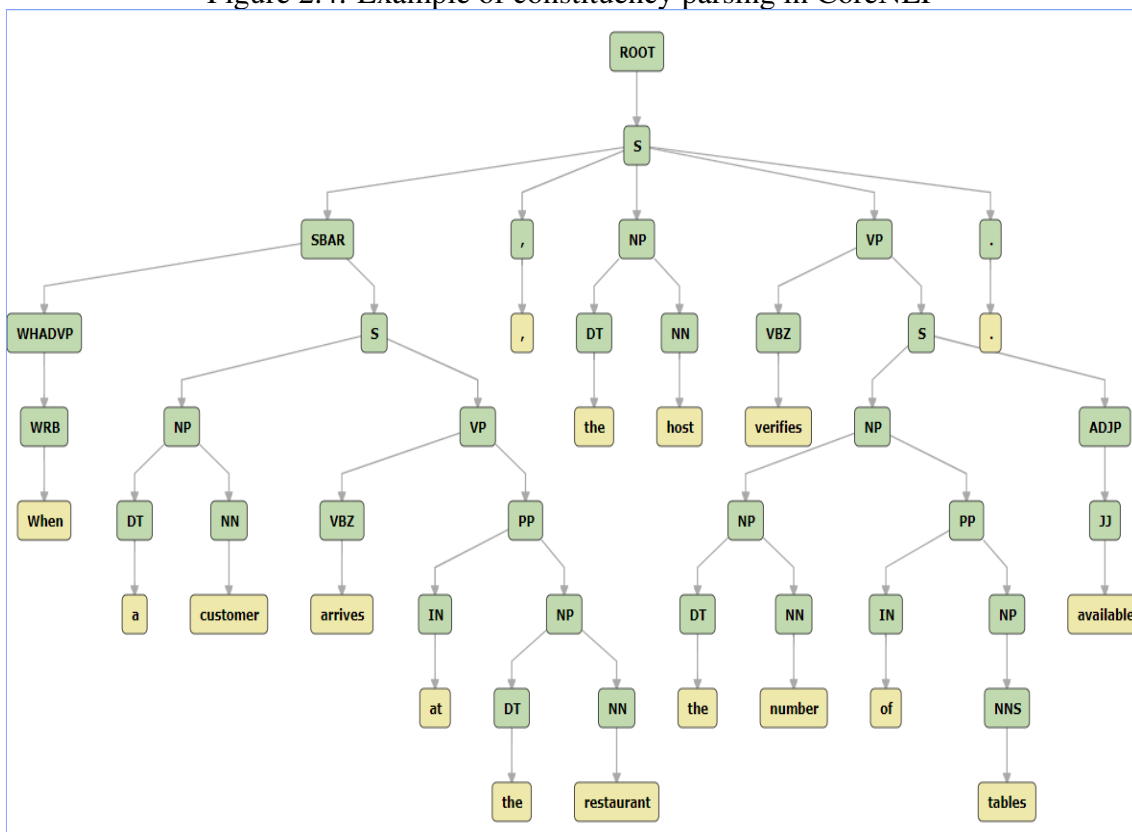
## 2.2.2 Stanford CoreNLP toolkit

The Stanford CoreNLP is a widely used implementation of the NLP pipeline (MANNING et al., 2014). Designed as framework in the JAVA programming language, the toolkit provides the core stages necessary for text analysis. In the context of the CoreNLP, each stage of the pipeline is referred as an *annotator* and can be included or omitted from the pipeline depending on the goal desired. As the toolkit follows the pipeline concept, some annotators require the presence of previous annotators in order to be executed.

Built based on the Stanford Parser, the Stanford CoreNLP toolkit allows the flexibility of choosing the parsing models for the constituency parsing and the dependency parsing. Such models may have different techniques on how to parse a given text. Those techniques include rule-based parsing, statistical parsing, neural-networking parsing and others which are available for the English language. Furthermore, some models are available for a number of languages.

One of the key contributions of the Stanford CoreNLP evolution is the capability

Figure 2.4: Example of constituency parsing in CoreNLP



Source: The Authors. Created with the Stanford CoreNLP Client (MANNING et al., 2014)

of establishing a web server with web services which facilitates the integration with other technologies. This interaction is further detailed in Chapter 5. Additionally, it also provides a web client interface in order to select and execute a NLP pipeline on a given input text. As example, Figure 2.4 shows the constituency parsing of the first sentence of the business process presented in Section 2.1 using the web interface available.

### 2.3 Extraction of business process elements

Driven by the observation that strictly manual process discovery methods can become a time-consuming task for large scale organizations (DUMAS et al., 2018), many authors have addressed this issue by introducing automated, or semi-automated, approaches for business process discovery. The input and output of the designed approaches vary from one research to another, however, this section aims to provide a background of a particular study used as basis for the approach presented in this study.

In the field of automated generation of business process models (i.e., output) from natural language text (i.e., input), the approach proposed by Friedrich, Mendling and Puhmann (2011) is considered the state of the art by recent studies (RIEFER; TERNIS; THALER, 2016). Despite having a distinct end goal, this study adapts and uses the technique for extraction of process elements from process descriptions. While Friedrich's approach seeks to generate a business process model following the BPMN, the approach presented in this study is focused on visually enhancing the process description as an artefact for further consumption.

A successful automated identification of process elements in natural language text is complex and relies on a well written text (i.e., grammatically correct) process descrip-

#### Listing 1: Input text rules

- The text actually describes a process and not, e.g., data objects or organizational structure.
- The text contains no questions.
- The text describes the process as perceived by an involved Actor and not on a meta level (“the next task is ...”, “then the process...”).
- The Process is described sequentially, meaning that the actions in adjacent sentences are related to each other.
- If a non sequential description is needed, the textual jump is made explicit in the text.

Source: (FRIEDRICH; MENDLING; PUHLMANN, 2011)

tion. Friedrich’s approach relies on base assumptions that should be respected in order to achieve a proper identification of process elements. Listing 1 shows the rules defined in the authors’ work.

Friedrich’s natural language analysis technique is divided in two levels: a *Sentence Level Analysis* and a *Text Level Analysis* (FRIEDRICH; MENDLING; PUHLMANN, 2011). In the first level, the analysis is performed separately on each individual sentence of the text. Using the Stanford Parser - a predecessor of the Stanford CoreNLP described in Section 2.2.2 -, the NLP pipeline is processed, resulting in split sentences along with their respective syntactical analysis (i.e., constituency and dependency parsing). Several algorithms are then applied, in order to extract sentence relevant information (e.g., actors, verbs and objects) and store in a data structure defined as *Action*. Every action instance relates to a verb identified in the sentence and additionally contains a set of properties encompassing a number of word dependencies, partially shown in Table 2.1. In some cases, a word dependency property might also be set to specific keywords.

Table 2.1: Main properties of the action data structure

Property	Description	Type
Name	Describes the verb as it appears on the text	String
Word index	Index of the verb in the sentence	Integer
Base form	Describes the verb’s plain form	String
Actor	Actor involved in the action	Actor data structure
Object	Object involved in the action	Object data structure
Specifiers	Complementary information in the sentence	Array of Specifier data structure
Det	Determiner of the head of a noun phrase (e.g., “the”, “which”)	Stanford dependency
Cop	Describes the copular verb of a clause (e.g., “is”, “are”)	Stanford dependency
Aux	Describes the non-main verb of a clause (e.g., “be”, “has”)	Stanford dependency
Mark	Introduces a clause subordinate to another clause (e.g., “wheter”, “although”)	Stanford dependency
	Condition indicators (“while” or “if”)	String
PreAdvMod	Modifies the meaning of an adverb	Stanford dependency
	Sequence indicators (e.g., “then”, “after”)	String

Source: The Authors

In the second level, the remaining unset properties of the actions previously created are filled with the information that could only be obtained by a holistic analysis of the text. Then, a link between the decoupled actions is created through a data structure defined as *Flow*. The flow also contains an indicator of which type of link was derived from the respective connected actions. In essence, a flow is responsible for connecting two or more actions and is categorized by one of five types: *sequential* flow, containing two actions performed in sequence; *choice* flow, containing actions included in an exclusive decision; *concurrency* flow, containing actions included in a parallel execution; *multiple choice* flow, containing actions included in an inclusive decision; *jump* flow, representing an explicit non sequential execution.

In Friedrich's approach, the steps mentioned above are succeeded by a *Process Model Generation* technique, which diverges from the goal of the approach designed in this study and, thus, not exploited.

## **2.4 Human-computer interaction**

One key standout contribution of the approach presented in this study is regarding visualization and user-interactive features applied to process descriptions. According to Sears and Jacko (2009), the study of human-computer interaction (HCI) concerns the interaction with systems which are capable of displaying, storing, processing and controlling information. When interacting with a computer, the user has specific goals and subgoals in mind. In order to accomplish these goals, the user initiates the interaction by giving commands (e.g., mouse clicks, text typing) for which the computer must handle, process and output an appropriate response.

Furthermore, the authors state that vision is the dominant modality of information transfer in HCI, which leads to an emphasis of researches in the stream of *visual selective attention*. Most importantly, the attention is the collection of processes which allow humans to dedicate their limited information processing capacity to a subset of information, through cognitive manipulation. As such, the information enters into the working memory and achieves the level of consciousness (SEARS; JACKO, 2009).

Therefore, capturing a human's attention through visualization features is a form of guiding the focus towards specific parts of information which are most important in a given context. In the context of this study, visualization features are used for highlighting a subset of BPMN 2.0 process elements present in a given process description, such that

the user's information processing emphasizes on specific chunks of texts which indicate relevant business process information.

While visualization features are a relevant factor for capturing a user's attention, a dynamic change in the environment is an effective manner of causing a *shift of attention*. Techniques such as the sudden and abrupt appearance or disappearance of a stimulus or a change in the color of a stimulus cause a shift of attention which can be used to quickly guide the user's attention towards the location of a specific important information (SEARS; JACKO, 2009). In the context of the approach present in this study, the shifts of attention are triggered by user initiated clicks, meaning that they fall in the category of action-centered attention.

The approach presented in this study is flexible and does not limit to any specific techniques for capturing the user's attention. However, Chapter 5 demonstrates a practical application of the approach, using *colors* and *icons* as visualization features applied in given process descriptions. The shifts of attention are triggered by mouse clicks on buttons and mouse hovering over chunks of texts. As such, the user initiates the interaction by clicking on buttons and the system handles the action by changing the color and adding BPMN 2.0 icons to the outputted process description, according to the selected process element type. This system response is defined as the annotation of a process element.

As per Shneiderman et al. (2016), the human visual system responds differently to various colors and different people may prefer or struggle with different spectrum of colors, either due to deficiencies or personal preference. According to the authors, as part of the principles of designing an effective HCI interface, blending interaction styles may be appropriate for natural language based interfaces when users are diverse. As such, the prototype developed in this study also covers the user interaction of modifying the colors of each process element type. Furthermore, a second interaction occurs when the user hovers the mouse through an annotated chunk of text, the system responds with a popup containing additional information about the respective process element.

### 3 RELATED WORKS

In order to reduce the effort required to discover and understand existing business processes in organizations, several authors have contributed in the field of facilitating the process discovery phase. Considering the wide range of contributions in this field, the related works were divided into two categories. The first category is the *automated extraction of business process information*, for which the relevant approaches are based on textual resources and may present different final goals. The second category focuses on the *generation of business process descriptions*, with emphasis on approaches having the similar goal as the approach presented in this study.

#### 3.1 Automated extraction of business process information

This section discusses the related works in the category of automated - or semi-automated - extraction of relevant business process information. The approaches included in this stream focus on analyzing existing textual resources in the organization with the purpose of helping in the discovery of processes. Table 3.1 shows recent related studies which fit in such category.

Table 3.1: Automated extraction of business process information

Authors	Approach
(FRIEDRICH; MENDLING; PUHLMANN, 2011)	Generate process model from natural language
(FERREIRA et al., 2017)	Identify process elements in natural language
(HONKISZ; KLUZA; WIŚNIEWSKI, 2018)	Generate process model from natural language
(LEOPOLD et al., 2019)	Search textual and model descriptions
(QIAN et al., 2020)	Generate process model from process texts
(IVANCHIKJ; SERBOUT; PAUTASSO, 2020)	Generate process model from domain specific language

Source: The Authors

In particular, process model extraction (PME) has been a focus topic addressed by many authors over the years. A number of studies which extract process information from textual resources share a common goal of automating the process model generation.



These approaches identify the process elements contained in a given text resource through the usage of NLP techniques and each author presents its own methodology for generating the process model. For instance, the work of Friedrich, Mendling and Puhmann (2011), detailed in Section 2.3, is an example of a study in this research topic and its approach for extracting business process information is used as basis for this study.

The study of Ferreira et al. (2017) aims to identify business process elements in natural language texts. First, a syntactic analysis parsing is performed to tag each sentence of the input text in relation to their part of speech (e.g., verb, subject, object etc.). Afterwards, each sentence is matched to one of 32 *mapping rules*, in which different syntactic structures are represented. The mapping rules are categorized and grouped in relation to a sub-set of BPMN 2.0 elements: activities, events, gateways and swimlanes. The last step is the output of the text mapped by the rules. In comparison to Ferreira's approach, this study also contains a phase for the identification of process elements, however, it does not rely on predefined syntactic structures, but rather on the detection of key words on the process description. As such, it is possible to detect the presence of process elements in any syntactic structure, including those not predicted by Ferreira's approach. In addition, this study goes beyond the scope of identification, by also annotating the process elements on the text with interactive visual features.

In the process model extraction research topic, the study of Honkisz, Kluza and Wiśniewski (2018) proposes an approach based on the syntactic analysis of natural language texts by extracting Subject-Verb-Object (SVO) constructs, which are later transformed into process activities. The authors divide the approach into five steps, in which the three initial steps are focused on the actual extraction of the business process information and the last two aim to generate an intermediate spreadsheet-based model and a BPMN model, respectively. In the context of information extraction, the initial step is to identify the possible process participants in each sentence. Following, the second step searches each sentence for the presence of Subject-Verb-Object constructs. In the last step, the text is further analyzed in the search of keywords indicating the presence of decision and parallel gateways. In comparison to the extraction phase of this study, both present a similar approach for identifying possible indicators of process elements based on the dependencies between each word in the text, obtained by the NLP dependency parsing. Unlike Honkisz's approach, however, the extraction phase in this study also accounts for the NLP constituency parsing, enabling a split of the sentence into sub-sentences in order to achieve a more granular analysis. As such, word dependencies between different

sub-sentences are inspected in a later step, during the holistic analysis of the text.

Leopold et al. (2019) proposes a structured and unified manner of extracting existing business process information from both process descriptions and process models. In order to achieve this, the authors define an unified data format which can store information from either process representation. The extraction is based on two different parsers: the *text parser*, for process descriptions; and the *model parser*, for process models. The text parser is particularly relevant as it is comparable to the extraction approach of this study. In terms of NLP techniques, both approaches are based on the pipeline of the Stanford Parser, using the dependency parsing to obtain the relation between words in the sentence. The approach to be presented in this study, however, uses the evolved Stanford CoreNLP version, introduced in Section 2.2.2. Regarding the actual analysis of text information, Leopold's approach also extracts information surrounding the verbs of the sentence, including subjects, objects and adverbs in order to create *Activity records*, similarly to the *Action* data structure presented in Section 2.3. The end goal of the information extraction in Leopold's approach, however, differs from this study, as it focuses on mapping verbs found in sentences to activity records following the RDF specification, rather than identifying the business process elements present in the sentences.

A recent study from Qian et al. (2020) contributing to the topic of process model extraction splits the text identification problem in three main steps: *sentence classification*, *sentence semantics analysis* and *semantic role labeling*. The first is responsible for identifying whether a sentence describes an action or a statement. Following, the second analyses the semantics of the sentence in order to identify the control of execution, such as successive relation, optional relation, concurrency relation. This step is conceptually similar to the creation of Flows presented in Section 2.3. Last, each word or phrase is assigned to a semantic role. In order to perform these steps, the authors propose a multi-grained text classifier leveraging neural networks trained for each specific step. In this paper, the authors focus on the machine learning technique rather than the extraction of the process elements. In that regard, their approach is considerably distinct, as a neural network classifier is used, while the approach to be presented uses a statistical classifier.

The study of Ivanchikj, Serbout and Pautasso (2020) proposes a framework to assist the process modeling phase by using a domain specific language as input and producing a BPMN model as the output. The authors introduce a strict language syntax for expressing the process description, facilitating the parsing technique to extract the process elements. Furthermore, no machine learning based techniques are mentioned. While the

approach presented in this study also contains a manual phase for preprocessing the natural language text, it does not define an entire new syntax but rather introduces keywords in the process description, which in several cases, are already present in the unprocessed text. Therefore, the manual effort required to create the domain specific language text is a considerable drawback factor in the analyzed study.

Even though the studies presented in this section achieve the purpose of automatically extracting relevant business process information, a number of them use such information to generate business process models. The remaining studies outside of the process model extraction topic, on the other hand, aim to identify the process elements either for unifying business process information or to be used as input for further approaches. In the context of this study, the extraction of process information is used to enhance existing organizational documentation in the form of process descriptions, providing a visual interactive usage which can be leveraged to accomplish different end goals.

### 3.2 Generation of business process descriptions

This section discussed the related works in the category of generation of business process descriptions. While a number of those studies accomplish this goal for different purposes, all of them present an automated generation of business process descriptions as an output. Furthermore, the approaches considered for this analysis are based on process models and natural language texts as inputs. Table 3.2 shows recent related studies which fit in this category.

Table 3.2: Generation of business process descriptions

- 
- From process models
    - (MALIK; BAJWA, 2012)
    - (LEOPOLD; MENDLING; POLYVYANYYY, 2012)
    - (MEITZ; LEOPOLD; MENDLING, 2013)
    - (RODRIGUES; AZEVEDO; REVOREDO, 2016)
    - (AYSOLMAZ et al., 2018)
- 
- From process models and natural language text
    - (SILVA et al., 2019)
    - (ZENG et al., 2020)
- 

Source: The Authors

Approaches in the stream of generation of business process descriptions based

exclusively on the process model create an output text depending on labels set for the process elements in the process model. Even though this can generate a more objective process description, it may also lack detailed information required for business analysts to understand the complete process and for employees to perform their activities. In general, studies in this stream aim to use the generated process description in order to perform an additional task, such as validating the business process model.

The study of Malik and Bajwa (2012) presents an approach to transform BPMN based models to business rules, which are represented by natural language texts. The approach is consisted of several steps, including the mapping of Flow Objects to predefined sentence templates for each process element type. Once identified, the connectors are used in order to establish an order to the previously created sentences. Additionally, Swim lanes and artefacts are also identified for grouping and adding additional information to sentences, respectively. The final steps focus on resolving dependencies and optimizing the natural language representation. As the approach to be presented is based on the original process description rather than predefined sentence templates, the generated process description includes more details, while still indicating the presence of the process elements.

Since the definition of the pipeline concept from natural language generation systems by Reiter and Dale (2000), text generation approaches have emerged following a common architecture. While the approach proposed by Leopold, Mendling and Polyvyanyy (2012) uses process models following the BPMN, a similar approach is presented by Meitz, Leopold and Mendling (2013), using process models in the form of Petri Nets. The natural language generation architecture approach is composed by three stages: *text planning*, *sentence planning* and *realization*. While these approaches have presented successful results regarding the completeness and structure of the generated process description, unlike the approach to be presented in this study, they do not propose any techniques for linking the generated output with the originator model.

The work of Rodrigues, Azevedo and Revoredo (2016) aims towards a more practical approach, presenting a framework following the pipeline concept for natural language generation systems. This particular tool stands out from previous approaches as it introduces a generic language independent component which can be extended for different language-specific implementations. The standard implementation is capable of generate texts based on BPMN models written in Portuguese and English. A basis motivation mentioned by the authors is the fact that domain experts might now always have knowledge

about the BPMN, however, the generated text still includes a certain level of technical terminology. In comparison, the approach to be presented is based on the original process description and visually indicates the presence of process elements, facilitating the understanding for readers not specialized with the modeling notation.

The study of Aysolmaz et al. (2018) defines a semi-automated approach for generating natural language requirements documents based on business process models. Unlike the previous analyzed works, this approach focuses on identifying activities on the process models which can either be supported by a system or fully automated by one, defined as “*automatable activities*”. The second step is to specify how the activities should be executed following a four-part investigation. Once the automatable activities are specified, the generation phase takes place to actually create the natural language requirements. Although text is indeed generated, this approach has a specific focus and does not aim to represent the complete business process. Furthermore, no visualization features are proposed, distinct to the approach to be proposed in this study.

Silva et al. (2019) proposes a service-oriented architecture for generating sound process descriptions. A sound process description is defined as structured, unambiguous, reveals possible quality and soundness problems related to BPMN 2.0, and contains clear identifiers for all known process elements in the original text. The approach comprises five services following the SOA principles (ERL, 2008). The *Main Service* is responsible for orchestrating the requests and responses between all the other services, while the Service Registry serves as an indexer, maintaining the location of the available services. The remaining three services: *Text Reader Service*, *Process Verification Service* and *Text Writer Service* represent each step of the approach to generate the sound process description. By leveraging the previous works of Friedrich, Mendling and Puhmann (2011), Ferreira et al. (2017), Leopold, Mendling and Polyvyanyy (2012) and the guidelines of process modeling (MENDLING; REIJERS; AALST, 2010), this approach combines each of those studies, thus, being able to generate process models, sound process descriptions and identify issues in process models, based on either a given natural language text or process model.

In comparison to Silva’s study, the approach to be presented is also based on the work of Friedrich, Mendling and Puhmann (2011). The Text Reader Service leverages from a very similar implementation in order to generate a process model from a process description. In this study, however, a re-implementation of Friedrich’s approach is performed and incremented with the purpose of identifying process elements in the original

process description. Moreover, the basis concept of *snippets*, introduced by Silva as a structured format for representing process elements as sequences of words in the text, is also adapted. In Section 4.4, a redesign of the snippet structure is presented. Unlike Silva's approach, however, the text annotation approach is performed on the given process description, instead of the generated sound process description. Additionally, this study is focused on the visual display of process elements on the text by not only adopting more visual features but also providing more flexibility (e.g., selection of element type color) and interaction (e.g., selection of annotated element types).

A recent study presented by Zeng et al. (2020) proposes an approach leveraging both process models and activity description templates with the purpose of repairing missing procedural texts (i.e., a series of simple activity descriptions). The approach begins processing the procedural text and the extracting information from the process model in order to perform a diagnoses of the missing activities on the text. At a later stage, the information extracted from the process model is used in order to select the templates for activities, which is a file generated from multiple textual descriptions of the model. The results of this stage are then used in order to repair the identified missing information in the procedural text. This particular approach implicitly tackles the issue raised when generating text exclusively based on process models, which in general, is the lack of detailed information. Although leveraging both existing process models and process descriptions, no link is performed between process elements and the generated text, unlike the approach presented in this study, which creates a visual representation of the process elements in the generated text.

In order to summarize the comparison of every analyzed work with the approach presented in this study, Table 3.3 presents each approach in respect to what it proposes to achieve, following five categories.

Table 3.3: Comparison of related works with the approach to be presented in this study

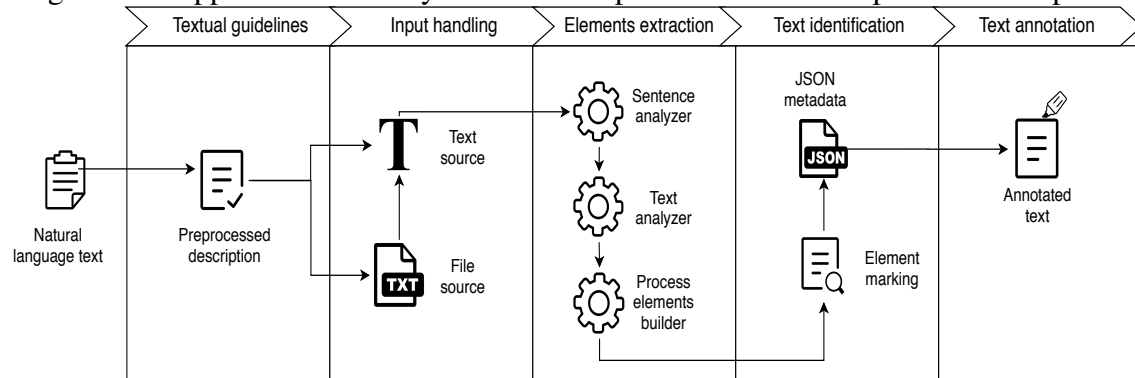
Authors	EPI	PME	TG	VAT	IGT
(FRIEDRICH; MENDLING; PUHLMANN, 2011)	X	X			
(MALIK; BAJWA, 2012)	X		X		
(LEOPOLD; MENDLING; POLYVYANYYY, 2012)	X		X		
(MEITZ; LEOPOLD; MENDLING, 2013)	X		X		
(RODRIGUES; AZEVEDO; REVOREDO, 2016)	X		X		
(FERREIRA et al., 2017)	X				
(AYSOLMAZ et al., 2018)	X		X		
(HONKISZ; KLUZA; WIŚNIEWSKI, 2018)	X	X			
(LEOPOLD et al., 2019)	X				
(SILVA et al., 2019)	X	X	X	X	
(QIAN et al., 2020)	X	X			
(ZENG et al., 2020)	X		X		
(IVANCHIKJ; SERBOUT; PAUTASSO, 2020)	X	X			
<b>Approach to be presented in this study</b>	X		X	X	X

Source: The Authors. Abbreviations: EPI - Extraction of Business Process Information, PME - Process Model Extraction, TG - Text Generation, VAT - Visual Annotation of Process Elements in Texts, IGT - User Interaction with Generated Texts

## 4 IDENTIFICATION AND ANNOTATION OF BUSINESS PROCESS ELEMENTS

This chapter details an user-interactive and visual approach for identification and annotations of business process elements in process descriptions. Fig. 4.1 shows the approach split into five phases. The first phase, *Textual guidelines*, defines a guide for writing a process description based on a natural language text, such that it becomes best suitable for the following phases. The *Input handling* serves as an intermediary phase responsible for preparing the data format of a given process description. Next, the *Elements extraction* phase performs the NLP techniques and algorithms for creating BPMN 2.0 data structures. Following, the *Text identification* phase searches for sequences of words which define each extracted element in the previous phase. Finally, the *Text annotation* phase is responsible for adding visual features in the process description. This approach focuses on the subset of BPMN 2.0 elements shown in Fig. 2.1. This particular subset consists of elements which are recurrently used in practice (e.g., Tasks, Start Events, Intermediate Events, End Events, Gateways) (MUEHLEN; RECKER, 2013).

Figure 4.1: Approach to identify and annotate process elements in process descriptions



Source: The Authors

### 4.1 Textual guidelines

As many sources of natural language documents describing business processes are non-standardized and written with no prior knowledge of how to properly describe it (SILVA et al., 2018), the first phase of the approach is a manual rewrite of the natural language text. To effectively automate the extraction of BPMN 2.0 elements from text sources, the process description should follow guidelines on how to express the tasks, events and decision points that comprises the business process. The rewritten process



description following the guidelines in this section is defined as the *preprocessed description*.

In order to express a business process as a preprocessed description in a proper way for the subsequent extraction phase, the first step is to define how to indicate the presence of each business process ingredient (i.e., tasks, events and decision points) introduced in Section 2.1. Tasks and events are generally detected in sentences with a syntactic construction containing verbs. In the case of events and decision points, key words are also used to explicitly evidenciate their existence in the preprocessed description. The particular key words selected are based on the basis study of Friedrich, Mendling and Puhmann (2011) introduced in Section 2.3, as this approach leverages an adaption of its technique for extracting the business process elements from the text.

*Tasks* should be expressed in the text using a verb that represents the action being performed, a subject, indicating the actor of this action and, when applicable, an object. Verbs in the process description must not be ambiguous in relation to its part of speech, as it can mislead the NLP model to incorrectly classify these words and, thus, preventing the algorithms in the extraction phase (sec. 4.3) from effectively identifying the necessary words that, eventually, will form a process element. Hence, ambiguous words should be replaced by synonyms which have a single part of speech class.

As tasks, *events* are also expressed through the presence of a verb in the sentence. As shown in Section 2.1, events may appear in process description in different verb tenses. Thus, in order to be distinguished from tasks, events should be preceded by conjunctions that indicate the property of temporal atomicity of the action. For such reason, the following conjunctions were defined as indicators of events: “*when*”, “*whenever*”, “*once*”, “*as soon as*”, “*after*”.

*Decision points*, on the other hand, do not require an specific syntactic construction in the sentences. Instead, they should be expressed by key words indicating a change in the control flow of the business process. These words, when followed by tasks or events, signalize a new branch of execution.

The following key words can be used in the process description preceding a task or event in order to indicate an exclusive decision: “*if*”, “*whether*”, “*in case of*”, “*in the case of*”, “*in case*”, “*for the case*”, “*whereas*”, “*otherwise*”, “*optionally*”. In particular, the “*while*” key word can also be used between two tasks for the same purpose, as exemplified in Table 4.1. Moreover, the conjunction “*or*” between two different subjects, verbs or objects is also a form to indicate exclusive branches for them. Similarly, parallel

executions should be indicated by the presence of: “*while*”, “*meanwhile*”, “*in parallel*”, “*concurrently*”, “*meantime*”, “*in the meantime*”. Additionally, the conjunction “and” between two or more different subjects also indicates parallel branches of a number of actors performing the same action. Finally, inclusive decisions should be expressed by using the conjunction “and/or” between different subjects, verbs or objects.

Table 4.1 demonstrates how the presented textual guidelines are used in order to adapt an existing unprocessed description to a preprocessed description. Each row of the table represents a sentence in both descriptions and contains the adaptations performed.

Table 4.1: Adaptation of a process description

Unprocessed sentence	Preprocessed sentence	Adaptations
A customer brings in a defective computer <del>and</del> the CRS <del>checks</del> the defect and hands out a repair cost calculation back.	<i>When</i> a customer brings in a defective computer, the CRS <i>verifies</i> the defect and hands out a repair cost calculation back.	Event key word Ambiguity removal
If the customer decides that the costs are acceptable, the process continues, otherwise she takes her computer home unrepaired.	If the customer decides that the costs are acceptable, the process continues, otherwise she takes her computer home unrepaired.	None
The ongoing repair consists of two activities, which are executed, in an arbitrary order.	The ongoing repair consists of two activities, which are executed, in an arbitrary order.	None, however, sentence not required
The first activity is to <del>check</del> and <del>repair</del> the hardware, <del>whereas</del> the second activity checks and configures the software.	The first activity is to <i>verify</i> and <i>adjust</i> the hardware <i>while</i> the second activity verifies and configures the software.	Ambiguity removal Exclusive decision key word
After each of these activities, the proper system functionality is tested.	After each of these activities, the proper system functionality is tested.	None
If an error is detected another arbitrary repair activity is executed, otherwise the repair is finished.	If an error is detected another arbitrary repair activity is executed, otherwise the repair is finished.	None

Source: The Authors

As further demonstrated in Chapter 6, the textual guidelines are not a mandatory requirement for the next phases of the approach. The unprocessed description is nevertheless a valid input for the approach, however, the extraction of the process elements is compromised, jeopardizing the subsequent phases, hence, providing a less accurate result.

## 4.2 Input handling

The next step is to use the preprocessed description as input for the further automated phases of the approach. This phase is responsible for establishing a web service, enabling the transference of the preprocessed description using the Hypertext Transfer Protocol (HTTP). This web service accepts HTTP requests using the POST method, which should contain the preprocessed description in the body of the request, either as a raw text content or as text file.

Additionally, this phase is responsible for handling the format of the content of the incoming requests in alignment with the expected format of the next phase. When a text file is received, its content is extracted and converted into a string format. The output of this phase is a string containing the preprocessed description.

## 4.3 Elements extraction

To effectively identify the process elements in the preprocessed description, the elements extraction phase adapts the approach proposed by Friedrich, Mendling and Puhmann (2011) of process model generation from NLP. As introduced in Section 2.3, Friedrich's approach can be divided in three main steps: *sentence level analysis*, *text level analysis* and process model generation, for which the latter was redesigned as the *process elements builder*. A number of modifications, however, have been performed based on Friedrich's prototype. Initially, deprecated technologies such as the Stanford Parser were readjusted in order to recreate a modern prototype further detailed in Chapter 5. Furthermore, implementation details of the each step of Friedrich's approach were also revised and are further explained in this section.

At the first stage, the whole text is submitted to the Stanford CoreNLP pipeline for the *constituency parsing* and the *dependency parsing*. The first uses a probabilistic trained model (KLEIN; MANNING, 2003) to generate a tree-like structure for each sentence, containing the part of speech of each token in it (e.g., subject, verb, adverb), while the latter extracts the Stanford dependencies (MARNEFFE et al., 2006) between each token of the sentence.

Next, each sentence is handled individually by the sentence analyzer, which is responsible for extracting the respective *actors*, *objects* and further complementary information (i.e., *specifiers*) involved around each *verb* identified in the sentence in order

to create the data structure *Action* encompassing all this information. Additionally, an action structure contains several properties derived from the dependency parse, meaning that it also stores the relation of the main verb to other words in the sentence, such as: adverbs, markers, prepositions, negations, auxiliaries and others. Furthermore, the sentence analyzer detects *conjunctions* between tokens, creating the data structure *ConjunctionElement* structure linking actors or actions. Each output of the sentence analyzer is an object of the class *AnalyzedSentence*, which contains all the actions and conjunctions extracted.

The main modifications performed in this step in comparison to the original approach are additions of properties in *Action* structure. The purpose of these properties are to store necessary information for the process elements builder step as well as for the further Text identification phase. Specifically, the properties *markerPos* and *realMarker* were included in order to store the position and the original key word that specifies a parallel or exclusive execution, respectively. These properties are useful to identify the start index of branches. Moreover, the existing property *preAdvMod* is complemented by an introduced boolean property *preAdvModFromSpec*. The former stores the adverb that precedes the action's verb, while the latter indicates whether the adverb is part of the action itself or part of the specifier. With this information, it is possible to enhance the logic of distinction between tasks and events.

After this step, the text analyzer consolidates the actions by resolving references, identifying markers and creating the data structure *Flow*, as detailed in Section 2.3. The flow type is defined based on the markers identified in each action as well as the conjunctions previously extracted in the sentence analyzer. Therefore, a process description following the guidelines presented in the previous phase is important because it enables a more accurate detection of key words, which are translated into markers.

In terms of adjustments performed in this step, a number of edge cases not handled by the original approach were covered. For example, when a split execution (i.e., decision point) is detected, the general logic is to create a join whenever a flow indicating the return to a sequential execution is analysed. However, if the text finishes during the split execution, in the original logic, the join flow would not be created, hence, additional logic had to be introduced to cover this. Furthermore, the possibility to detect the “*while*” marker in-between two actions and create a concurrency flow is also covered. Finally, the flow type *jump* was removed from the logic, as it is useful for the goal of Friedrich's approach of creating the process model, but not applicable for the goal of the approach in this study.

The process elements builder is the last step of the elements extraction. Unlike in Friedrich’s approach, this step does not generate a process model, but rather creates data structures that represent *BPMN 2.0 elements*. In order to achieve that, first, each flow is analysed and translated as one or more BPMN 2.0 element object (e.g., Tasks, Intermediate events, Gateways). Then, the elements are linked through a data structure representing *sequence flows*, enabling the graph-like representation of a process model. Last, particular elements are further inspected, such as elements whose sequence flows have no predecessor or successors, in order to identify Start or End events, as well as elements originated from actions which are not relevant for process modeling, hence, removable. The final output of the element extraction phase is a data structure, containing all nodes (elements) and edges (sequence flows) created during the process.

Besides the distinct output, the main contributions in the process elements builder in comparison to Friedrich’s approach is regarding the detection of process elements in the action structures. In the original logic, the distinction between a task and an event is purely based on the action’s marker property. Instead, the presented approach introduces a more granular verification which searches for key words that indicate finished actions in order to detect events. These key words are part of the textual guidelines defined in the previous section. Algorithm 1 shows the high level additional logic introduced for detecting events based on key words. This algorithm was developed to serve as a complementary logic and it is not present in the original approach. Furthermore, it was observed that timer events and tasks are recurrently present in the same action structure. Considering the original implementation, it was not possible to detect a timer event and a task from the same action, therefore, the logic for detecting timer events has been decoupled from the distinction between tasks and events and performed as a separate step.

---

**Algorithm 1:** Algorithm for detecting possible events

---

```

input : Action
output: Boolean
1 if ( $\text{action.preAdvMod} \wedge \neg \text{action.preAdvModFromSpec}$ )  $\vee$   $\text{action.marker}$  then
2    $\text{sentenceText} = \text{action.sentence}$ ;
3    $\text{minIndex} = \max(\text{action.preAdvPos}, \text{action.markerPos})$ ;
4   for  $\text{indicator}$  in  $\text{finishedIndicators}$  do
5      $\text{indicatorIndex} = \text{indexOf}(\text{sentenceText}, \text{indicator})$ ;
6     if  $\text{minIndex} \leq \text{indicatorIndex} \leq \text{action.wordIndex}$  then
7        $\text{return true}$ ;
8 return false;

```

---

#### 4.4 Text identification

Once the process model data structure is formed, the next task is to identify the sequence of words in the text that represent each process element in the process description and expose the results as a consumable web service. As the practical form of representing these sequences, Listing 2 demonstrates the redesigned data structure of *snippets*, based on the work of Silva et al. (2019). Each snippet contains the indices of the word that starts the sequence and the word that finishes it, a process element unique identifier, the type of the process element (e.g., Task, Conditional Event, XOR Gateway) and the identifier of the actor corresponding to the element.

Listing 2: JSON snippet structure

```
{
  "startIndex" : int,
  "endIndex" : int,
  "processElementId" : int,
  "processElementType" : string,
  "resourceId" : int
}
```

Algorithm 2 shows the method responsible for generating the output metadata of this phase. First, each process element receives its own unique identifier in order to facilitate the linking with their respective lanes. Second, the nodes of the process model are set through a filtering logic to avoid duplicated and overlapping snippets as well as a classification logic to categorize them as: *decision nodes*, *gateway nodes* or *regular nodes*. The filtering logic is detailed in Algorithm 3 and is necessary due to the observation that multiple process nodes can be extracted originating from the same chunk of text. For example, in the following sentence: “*Once the food, wine, and cart are ready, the waiter delivers it to the guest’s room*”, three different events can emerge from the first phrase, while sharing the same action (i.e., “*are ready*”). So, to avoid the mentioned issue, a single node is selected to represent the corresponding range of indices, while the others are filtered out. The classification, on the other hand, is important for two reasons: it separates the gateways from every other node, assisting in the next steps; it enables the identification of the decision nodes of each exclusive split gateway.

After this preparatory phase, the creation of the snippets is performed by analyzing the text element structure (i.e., action, actor, specifier, flow) that originated each process element, searching for the start and end words of the snippet, based on the given element

---

**Algorithm 2:** Algorithm for generating the JSON metadata
 

---

```

input : Process model
output: Structured metadata

1 CreateIds (processModel);
2 decisionActions, gatewayNodes, regularNodes = FilterElements
  (processModel);
3 resourceList = CreateResourceList (processModel);
4 text = CreateText (regularNodes);
5 gateways = CreateGateways (gatewayNodes, decisionActions);
6 return resourceList, text, gateways;

```

---



---

**Algorithm 3:** Algorithm for filtering and classifying process elements
 

---

```

input : Process model
output: Decision nodes, Gateway nodes, Regular nodes

1 for node in processNodes do
2   if IsInstance (node, Lane) then
3     regularNodes  $\cup$  node;
4   else if IsInstance (node, Task)  $\vee$  IsInstance (node, Event) then
5     nodeIndexMap [GetIndex (node)]  $\cup$  node;
6   else if IsInstance (node, Gateway) then
7     gatewayNodes  $\cup$  node;
8     if IsXor then
9       for branch in branches do
10        decisionActions  $\cup$  branch;
11 for nodeList in nodeIndexMap do
12   regularNodes  $\cup$  GetRepresentative (nodeList);
13 return regularNodes, gatewayNodes, decisionActions;

```

---

type. Algorithm 4 details how the start index is obtained using the properties stored in the element. Similarly, Algorithm 5 details the logic for detecting the end index. Each algorithm considers different types of word dependencies stored in the given element and appends them to a list with the possible candidates for the corresponding index. In particular, the detection of end indices must also account for the number of words in the name of the text element. For example, a specifier might contain several words that complement an action, therefore, the goal is to obtain the index of the last word, instead of the first.

The identification of gateways, on the other hand, requires a more complex logic. First, different algorithms have to be considered based on whether the process element is a split or a join gateway. In a split gateway, all branches must be accounted for, therefore, each position that contains a new branch must be identified. In a join gateway, on the

---

**Algorithm 4:** Algorithm for identifying the start index of elements
 

---

```

input : Element
output: Start Index

1 candidates = [];
2 if IsInstance (element, Actor)  $\vee$  IsInstance (element, Resource) then
3   if element.determiner then
4     determiner = 1;
5   else
6     determiner = 0;
7   candidates  $\cup$  (element.wordIndex – determiner);
8   for specifier in element.specifiers do
9     candidates  $\cup$  (specifier.worIndex – determiner);
10 else if IsInstance (element, Specifier) then
11   candidates  $\cup$  element.worIndex;
12 else if IsInstance (element, Action) then
13   candidates  $\cup$  element.worIndex;
14   if element.auxiliar then
15     candidates [0] – = 1;
16 return min (candidates, default=1);

```

---

other hand, only one position is relevant. A second important aspect of identifying a gateway is whether it is explicitly written on the text using the key worlds described in Section 4.1 or if it was implicitly detected. This information is also added to the metadata and further consumed in the Text annotation phase. The position of implicit branches of split gateways is detected by the presence of the conjunctions *and*, *or*, *and/or*, while the position of implicit join gateways is inferred based on the last branch found in the text.

Algorithm 6 demonstrates the logic for identifying indices in a split gateway *branch*. The first step is to check which type of gateway is being analyzed (i.e., exclusive, parallel or inclusive). Following, the second step is to verify if the text element which originated the branch contains any of the explicit indicators corresponding to the gateway type. In such case, the start and end indices can be determined based on the indicator position and by leveraging Algorithm 5, respectively. In case the text element does not contain any explicit indicators, the fallback logic is to identify which part of the text element is being connected by the conjunctions (i.e., and, or, and/or), which can be either the actor, the object or the action itself. Afterwards, the algorithms 4 and 5 are used to determine the start and end indices, respectively.

Similarly, Algorithm 7 shows the logic for determining indices in a join gateway. Unlike the previous algorithm, however, identifying the indices of a join does not depend



---

**Algorithm 5:** Algorithm for identifying the end index of elements
 

---

```

input : Element
output: End Index

1 candidates = [];
2 if IsInstance (element, Actor)  $\vee$  IsInstance (element, Resource) then
3   candidates  $\cup$  (element.wordIndex);
4   for specifier in element.specifiers do
5     candidates  $\cup$  (specifier.wordIndex);
6 else if IsInstance (element, Specifier) then
7   candidates  $\cup$  (element.wordIndex + count (element.name, " "));
8 else if IsInstance (element, Action) then
9   candidates  $\cup$  element.wordIndex;
10  if element.cop then
11    candidates  $\cup$  element.copIndex;
12  for specifier in element.specifiers do
13    if specifier.wordIndex > element.wordIndex then
14      candidates  $\cup$  (specifier.wordIndex + count (specifier.name, " "));
15  if element.object then
16    if element.object.wordIndex > element.wordIndex then
17      candidates  $\cup$  (element.object.wordIndex + count
18        (element.object.name, " "));
19      for specifier in element.specifiers do
20        if specifier.wordIndex > element.wordIndex then
21          candidates  $\cup$  (specifier.wordIndex + count
22            (specifier.name, " "));
21  if element.xcomp then
22    candidates  $\cup$  getElementEndIndex (element.xcomp);
23 return max (candidates, default=1);

```

---

on the gateway type. Instead, the approach is to detect which is the process element that succeeds the join and search for forward indicators in its corresponding sentence. In case an indicator is found and it is located in-between the index of the next element and the end index of the last branch of the gateway, then the indicator position is used for determining the start and end indices. In case there are no explicit indicators, on the other hand, the fallback logic is to utilize Algorithm 5 applied to the gateway's last branch found in the sentence in order to determine both the start and end indices.

Listing 3 shows the generated JSON structure containing the properties *resourceList*, *text* and *gateways*. The *resourceList* is a simple array-like structure with all the extracted lanes, containing their respective identifiers and labels. The *text* is a list of structures that represent each sentence. Each sentence structure is comprised by its unique identi-

---

**Algorithm 6:** Algorithm for identifying the indices of a split gateway branch
 

---

```

input : Gateway, Element
output: Start Index, End Index, Is Explicit

1 isExplicit = true;
2 if gateway.type == EXCLUSIVEGATEWAY then
3   if element.marker in conditionIndicators  $\wedge$  element.markerPos > 0 then
4     startIndex = element.markerPos;
5     if element in decisionActions then
6       endIndex = getElementEndIndex (element);
7     else if element.realMarker then
8       endIndex = startIndex + count (element.realMarker, " ");
9     else
10      endIndex = startIndex;
11  else if element.preAdvMod in conditionIndicators  $\wedge$ 
    element.preAdvModPos > 0 then
12    startIndex = element.preAdvModPos;
13    if element in decisionActions then
14      endIndex = getElementEndIndex (element);
15    else
16      endIndex = startIndex;
17 else if gateway.type == PARALLELGATEWAY then
18   if element.marker in parallelIndicators  $\wedge$  element.markerPos > 0 then
19     startIndex = element.markerPos;
20     if element.realMarker then
21       endIndex = startIndex + count (element.realMarker, " ");
22     else
23       endIndex = startIndex;
24 if  $\neg$ startIndex  $\vee$   $\neg$ endIndex then
25   elementToMark = findBranchElement (gateway, element);
26   startIndex = getElementStartIndex (elementToMark);
27   endIndex = getElementEndIndex (elementToMark);
28   isExplicit = false;
29 return startIndex, endIndex, isExplicit;

```

---

fier, the original string and a snippet list containing the regular nodes' information and indices. Unlike the text property, the gateways cannot simply be grouped by sentences, as it is possible that one or more branches of a gateway were originated from different sentences. Thus, each gateway is set as an independent structure and encompasses a list of its branches, containing their respective indices, similar to the sentence snippets, but with the addition of its sentence identifier as well as a boolean property indicating if the branch was detected based on an explicit indicator on the text. Finally, the completed JSON

**Algorithm 7:** Algorithm for identifying the indices of join gateways

---

```

input : Gateway
output: Next Element, Start Index, End Index, Is Explicit

1 isExplicit = false;
2 nextElement = getNextElement (gateway);
3 branches = sort (nextElement.multiples, reverse=true);
4 lastBranch = branches [0];
5 if nextElement.index > lastBranch.index then
6     sentence = nextElement.sentence;
7     for indicator in WordNetWrapper.acceptedForwardLinks do
8         indicatorIndex = indexOf (sentence, indicator);
9         if getElementEndIndex (lastBranch) < indicatorIndex <
            nextElement.wordIndex then
10            startIndex = indicatorIndex;
11            endIndex = indicatorIndex + count (indicator, " ");
12            isExplicit = true;
13 if ¬isExplicit then
14     nextElement = lastBranch;
15     startIndex = getElementEndIndex (nextElement) + 1;
16     endIndex = startIndex;
17 return nextElement, startIndex, endIndex, isExplicit;

```

---

Listing 3: JSON metadata structure

```

{
  "resourceList": [{
    "id" : int,
    "name": string
  }],
  "text" : [{
    "sentenceId" : int,
    "value" : string,
    "snippetList" : array<Snippet>
  }],
  "gateways" : [{
    "processElementId" : int,
    "processElementType" : string,
    "resourceId" : int,
    "branches" : array<Snippet>
  }]
}

```

structure is exposed as a web service, granting a flexibility for the identified elements to be displayed by different types of visualization features.

## 4.5 Text annotation

The last phase of the approach is to visually indicate the sections of the text that represent each of the process elements extracted and identified in the previous phases. The proposed approach focuses on mapping each type of identifiable BPMN 2.0 elements to a different color and to its respective symbol according to the official OMG specification (COMMISSION et al., 2013). To do so, the JSON metadata generated in the Text identification phase must be parsed in order to obtain the original text in addition to the snippets of each process element and branch. The result of this phase is an annotated process description.

Initially, the *resourceList* property is handled separately from the others. At this step, the labels of the lanes are directly parsed and displayed with no dependencies to the other properties of the metadata. Afterwards, the *text* and *gateways* properties of the metadata are merged. As demonstrated by Algorithm 8, the *text* property is iterated over, creating a mapping structure, for which the key is the sentence identifier and the value contains the original string of the sentence as well as the snippets of the process elements. Then, each gateway in the *gateways* property is analyzed by iterating over its branches. As each branch contains their respective sentence identifier, it is possible to append the gateway data in the previously created mapping structure. Thus, for each branch contained in the gateway, a snippet is added, including the boolean property *isBranch*, which indicates that this is not a BPMN 2.0 element, but rather a branch of one. At the end of this step, the mapping structure is a merge of the *text* and *gateways* properties.

---

### Algorithm 8: Algorithm for merging properties of the JSON metadata

---

```

input : JSON metadata
output: Sentence Map

1 sentenceMap = {};
2 for sentence in text do
3   | sentenceMap [sentenceId]  $\cup$  sentence.value;
4   | sentenceMap [sentenceId]  $\cup$  sentence.snippetList;
5 for gateway in gateways do
6   | for branch in gateway.branches do
7     | snippet = CreateSnippet (gateway, branch, isBranch = true);
8     | sentenceMap [sentenceId].snippetList  $\cup$  snippet;
9 return sentenceMap;

```

---

The next step is to link each *word* of the text with its respective process element type. Algorithm 9 shows a high level logic for creating a mapping structure between

each word index present in the range of every snippet and a structure which contains the element type, resource id and, when applicable, the respective icon and preceding text. In order to achieve this, every snippet present in sentence map created in the previous step is iterated over. As this is an user-interactive approach, the first verification is to check if the snippet's element type is currently selected by the end user to be annotated. If so, the indices of the respective snippet are iterated over, from the start index until the end index. At this point, the structure is filled with a map between the word index and information obtained from the snippet. In case the snippet represents a process element, the respective process element type and resource id are mapped to the corresponding word index. In addition, if the analyzed word index corresponds to the start index of the snippet, the BPMN 2.0 symbol of the element type is also mapped. On the other hand, in case the snippet represents a branch, only a complementary text containing the type of the gateway is mapped.

---

**Algorithm 9:** Algorithm for assigning each word to its process element type

---

```

input : Sentence Map
output: Word Map

1 wordMap = {};
2 for sentence in sentenceMap do
3   for snippet in sentence.snippetList do
4     markerData = selectedMarkers [snippet.processElementType];
5     if markerData.checked then
6       for wordIndex in (snippet.startIndex, snippet.endIndex) do
7         if  $\neg$ snippet.isBranch  $\vee$  snippet.isExplicit then
8           wordMap [wordIndex] = {
9             elementType: snippet.processElementType,
10            resourceId: snippet.resourceId };
11          if wordIndex == snippet.startIndex then
12            wordMap [wordIndex].icon = markerData.icon;
13          if wordIndex == snippet.startIndex  $\wedge$  snippet.isBranch then
14            wordMap [wordIndex].preText =
15              snippet.processElementType;
16
17 return wordMap

```

---

The previous step created a mapping structure for which the keys are all the word indices obtained from the snippets created in Section 4.4. Such structure, however, still does not contain *every* word index of the entire text, but rather the ones which represent a process element. Therefore, in order to generate the entire original text in addition with the visual features, it is necessary to iterate over the *value* property, as it contains the entire

inputted text of the sentence. Algorithm 10 demonstrates how this is performed in order to generate an *annotated text list*. First, the original string of sentence is split into words, whose indices are iterated over. The initial color of a word is defined by a previously set default color. However, in case the analyzed word index is part of the mapping structure created in the previous step, the color is redefined based on the color of the corresponding word index process element type. Additionally, when applicable, the icon and preceding text are appended to the annotated text list. At last, the actual word and its color are appended to the list.

---

**Algorithm 10:** Algorithm for adding visual features in a text structure

---

```

input : Sentence, Word Map
output: Annotated Text

1 annotatedText = [];
2 sentenceWords = split (sentence.value, " ");
3 for wordIndex in sentenceWords do
4   color = DEFAULTCOLOR;
5   wordData = wordMap [wordIndex ];
6   if wordData then
7     if wordData.icon then
8       annotatedText  $\cup$  {icon: wordData.icon};
9     if wordData.preText then
10      annotatedText  $\cup$  {preText: wordData.preText};
11      color = selectedMarkers [wordData.elementType];
12  annotatedText  $\cup$  {color: color, word : sentenceWords [wordIndex ]};
13 return annotatedText;

```

---

Finally, after the annotated text list is created, every word in the text has its own color defined in addition to indicators of symbols - for the case of the first word which indicates process elements - as well as preceding texts - for the case of words which indicate branches. Based on such information, the actual user-interactive visual representation depends on the technology used. Chapter 5 presents the prototype created, which includes a possible implementation of the annotation approach presented in this section.

## 5 PROTOTYPE

This chapter details the development of a practical application of the approach presented in chapter 4. A prototype was developed in order to demonstrate the extend of the capabilities of the approach in a number of use cases for which it could be used.

First, the *architecture* of the prototype is established, defining the architectural pattern and the motivation behind it. Following on a more granular level, the *technologies* of each component of the prototype are defined, as well as the chosen programming languages and the reason why they are suitable for this context. Next, the implementation details of each component of the prototype are explained and additionally detailed leveraging UML class and sequence diagrams. Finally, the functioning prototype is demonstrated, along with a set of its features.

### 5.1 Architecture

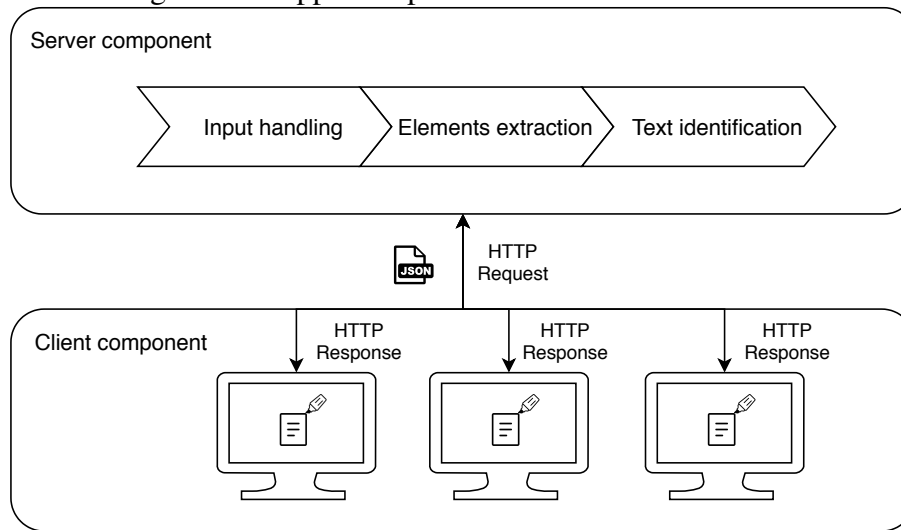
The first step for developing the prototype was the definition of the architecture. Establishing a suitable architecture enables a better choice of technologies which are capable of integrate with crucial dependencies of the approach, such as the NLP techniques. Furthermore, the chosen architecture must be able to support every automated phase of the presented approach and still enable the flexibility of the user interactive visualization approach.

A key motivation behind this prototype is to enable accessibility and reusability for users, therefore, a network-based architectural style is particularly suitable for this context. According to Alonso et al. (2004), web services are a way to expose the functionality of an information system and make it available through standard web technologies. Furthermore, Sheng et al. (2014) states that web services provide simple access to resources, making them accessible via the internet, enabling applications to combine and reuse them.

Considering the advantages of web services and the motivation presented in Chapter 1 to build an accessible prototype, this study presents a network client-server model based on web services for requesting and providing functionalities. In accordance to Fielding and Taylor (2000), in a client-server architecture, the client component sends requests to the server and waits for a response back. The server may either reject or process the requests.

Applying the client-server model to the approach presented in Section 4, the server

Figure 5.1: Approach phases in a client-server model



Source: The Authors

component is built in order to perform the phases between the *Input handling* until the *Text identification*. The Input handling is triggered by a request from the client side, and the Text identification exposes its output through the response of the request. The client component, on the other hand, is responsible for the last phase of the presented approach, the *Text annotation*. The communication and data transferring between the client and the server components is performed using the HTTP protocol. Furthermore, the data exchange format follows the widely used JSON format. Figure 5.1 shows the phases of the presented approach in relation to the chosen architecture.

## 5.2 Technologies

A key motivation for developing a prototype for the presented approach is due to the fact that many researches in the area of extracting process models and elements from business process descriptions are not possible to be reproduced due to technological incompatibilities or lack of proper documentation of their respective prototypes. Considering this reason, the technologies for the development of this prototype were chosen with the intend of providing a reproducible and incremental study, while maintaining the approaches used as basis for this study.

According to Oliphant (2007), the Python programming language excels as a platform for scientific computing, due to reasons such as: an open source license to use, distribute any Python-based applications; no concerns about portability, as it runs on several platforms; clear syntax and sophisticated constructs enabling procedural or object-



oriented codes; ability to interact with a wide variety of other software components. Furthermore, Millman and Aivazis (2011) states that Python has become the standard language for exploratory, interactive and computation-driven research. Based on those considerations, Python was chosen as the programming language for the server component of our architecture. This implicates, however, that the code developed by Friedrich, Mendling and Puhmann (2011) and Silva et al. (2019) had to be adapted to a different programming language, as they were originally developed in Java. In the next sections of this chapter, it is shown how the previous researches code was adapted and further expanded by our own methodology for identifying the business process elements in the text.

Shifting towards a different programming language is also challenging in terms of libraries and frameworks usage. The basis Java code allowed a simple interaction with the also Java written software *Stanford Parser* for the NLP phase, while a Python code must handle an integration between both programming languages. Additionally, since the original study by Friedrich et. al., the Stanford Parser has evolved into the *Stanford CoreNLP* software, introduced in section 2.2.2. Python enables the interaction with several software components, specially when supported by its own libraries.

The NLP pipeline is a key functionality of the presented approach and is handled by server component. Covered by the *Stanford CoreNLP* software, it enables the logic that follows in the Elements extractions phase. The CoreNLP software, however, is a Java based application, which requires a special handling on the server component's side. This cross programming language integration is assisted by Python's Natural Language Toolkit library suite (BIRD, 2009), which contains its own module and logic in order to enable the interaction with the CoreNLP pipeline. The mentioned module leverages the CoreNLP Server - embedded in the CoreNLP software -, which can be instantiated as a local web server and provides a web API using the HTTP protocol, allowing the usage of every functionality available in the toolkit.

The client component, on the other hand, has the responsibility of covering the *Text annotation* phase of the presented approach in a web scenario, therefore, a mandatory requirement is enabling end-users to access, visualize, interact and analyse data. As per Heineman and Council (2001), JavaScript provides the ability to animate a web page for user interaction (e.g., mouse click, keyboard entries), as well as the ability to access programs running on a server. Furthermore, currently, the vast majority of browsers support JavaScript, hence, portability is not a concern. Also, according to Shute (2019),

JavaScript is one of the backbones of web development and nearly every major website uses it. Motivated by the requirements and the presented capabilities, JavaScript was chosen as the programming language for the client component.

In order to achieve complex visual features on a web page, while maintaining a clean code structure, a JavaScript based web application can be enhanced with the usage of frameworks and libraries. The state of the art *ReactJS* library was used for building modular user interfaces using a Single Page Application (SPA). As per Vipul and Sonpatki (2016), the SPA principle enables web applications which do not require a complete reload of the page in order to perform a change in the display or perform a user interaction. According to Aggarwal (2018), ReactJS shows a high performance in comparison to other widely used frameworks and libraries mainly due to the modifications performed first on a virtual Document Object Model (DOM), followed by an alignment with the browser's DOM, as opposed to the standard approach of directly modifying the browser's DOM.

In comparison, the study of Silva et al. (2019) combined Friedrich's process model extraction approach with the text generation approach of Leopold et al. Considering that both Friedrich's and Leopold's prototypes were developed using the JAVA programming language, it was suitable for Silva's prototype to also use the same programming language, as it enabled a straightforward integration between them. On the other hand, Silva's prototype already leveraged JavaScript for the visualization features.

Even though the approach presented in this study also leverages the text processing techniques of Friedrich's work, the prototype presented in this study shifts towards modern technologies, both in the context of the text processing and in the context of display of information. The JAVA based text processing in Friedrich's and Silva's prototypes was transposed and adapted to the Python programming language, enabling the flexibility to upgrade from the Stanford Parser to the Stanford CoreNLP. Furthermore, while the visualization features in Silva's prototype already leveraged web technologies, the prototype presented in this study introduces the usage of ReactJS, enabling the integration with several existing modular components for visualization and interaction features.

### **5.3 Server component**

The server component was structured in accordance with the object-oriented paradigm following the Separation of Concerns (SoC) design principle. According to Laplante

(2007), a modular design encapsulates the software behaviour in software units and is a requirement in order to achieve a clear Separation of Concerns in an object-oriented design. Additionally, Laplante also states that a modular design leads to a high cohesion, where each module aims to solve a single part of the overall problem. As such, in order to reduce the coupling between classes, data-centered classes were designed to store and transfer data which are accessed and updated throughout multiple steps of the application.

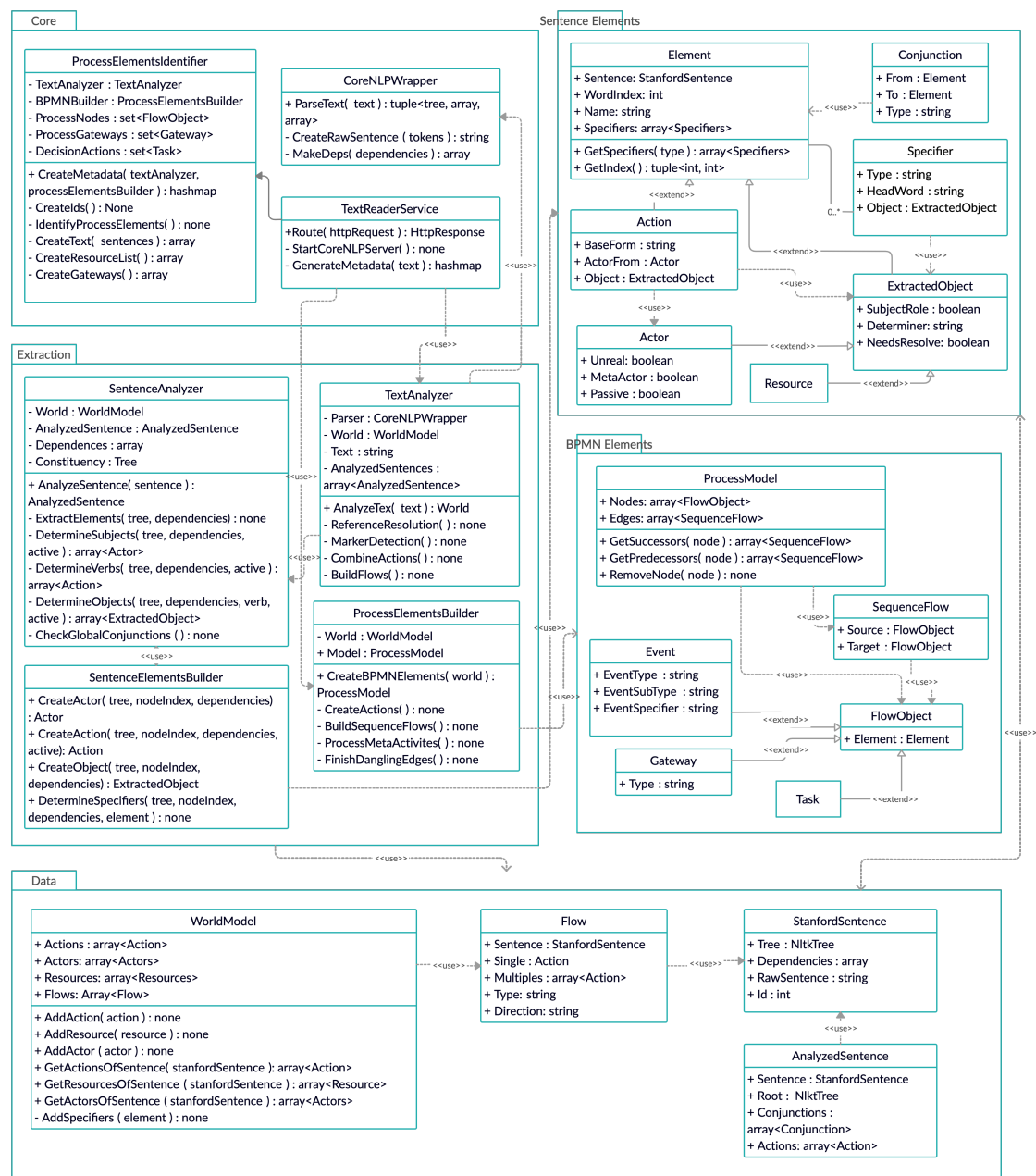
In that sense, classes of the server component can be categorized either as a *logic-centered* or as a *data-centered* classes. Logic-centered classes contain algorithms directly connected with the approach described in Chapter 4. Essentially, these classes represent either a phase, a step of a phase, or a core functionality of a step (e.g., CoreNLP communication interface). Data structure level classes, on the other hand, contain methods designed to maintain and manipulate data which is reused across multiple logic-centered classes. Nevertheless, every class is composed by methods and properties relative to their own behaviour, regardless of its category. Furthermore, a *utility* package was created in order to store recurrently used functions and global constants.

Figure 5.2 shows the class diagram of the server component. The designed architecture establishes a Separation of Concerns between each step of the presented approach. As such, the classes in the *Extraction* package were built in order to reflect the Elements extraction phase and its logic is reinterpreted and adapted from the Java code of the basis work of Friedrich et. al. to the Python programming language. The *ProcessElementsBuilder* class, however, is an exception, as its logic and goal is different from Friedrich's *ProcessModelBuilder* class. The former focuses on enhanced algorithms that leverage extracted information from the process description in order to build objects that represent process elements; the latter, on the other hand, has the goal of creating a visual representation in the form of a process model. Furthermore, the classes present in the *SentenceElements* and *Data* packages are reinterpreted data structures of Friedrich's work and are used throughout every step of this phase.

As the entry point of the server component, the *TextReaderService* serves as the orchestrator between the logic-centered classes responsible for the phases of the approach. Therefore, it is directly connected with the *SentenceAnalyzer*, *TextAnalyzer*, *ProcessElementsBuilder* and *ProcessElementsIdentifier* classes. Table 5.1 shows how the server component relates some of its classes with the phases of the presented approach. Fundamentally, the design is as such that each step of a phase is handled by a separate class.

The *WorldModel* and *ProcessModel*, on the other hand, are the main data-centered

Figure 5.2: Class diagram of the server component



Source: The Authors

classes. During the Elements extraction phase, the core classes involved store data in the *WorldModel* class, designed as the central point of information for this phase. At the first step, the *WorldModel* instance contains the data generated by *SentenceAnalyzer* class, which, following the presented phase in Section 4.3, refers to lists of *Action*, *Actor* and *Resource* objects created based on retrieved information from the process description. In the next step, the *WorldModel* is updated with lists of *Flow* class objects, created by the logic of the *TextAnalyzer*. Consequently, the *WorldModel* instance contains all the information from the process description properly extracted and joined.

Table 5.1: Classes in relation to phases of the presented approach

Approach phase	Class	Package
Input handling	TextReaderService	Core
Elements extractions	SentenceAnalyzer	Extraction
	TextAnalyzer	Extraction
	ProcessElementsBuilder	Extraction
Text indetification	ProcessElementsIdentifier	Core

Source: The Authors

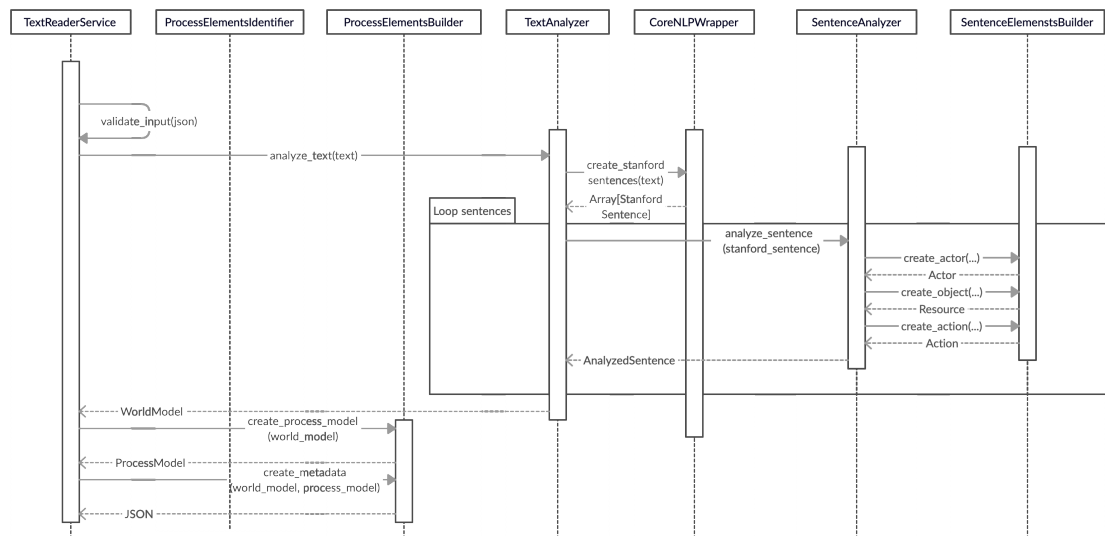
Finally, the *ProcessModel* class encapsulates the data created from the last step of the Elements extraction. The *ProcessElementsBuilder* class builds objects corresponding to the process elements identifiable by the approach, such as Tasks, Events and Gateways. Once completed, the instances of the *WorldModel* and the *ProcessModel* serve as inputs for the Text identification phase, which is the last phase covered by the server component.

Figure 5.3 shows the sequence diagram representing the complete flow of the server component. Once the *TextReaderService* is initiated, the *CoreNLP Server* is started in order to enable future requests to the CoreNLP pipeline. Once the initialization is ready, every post request received by the *TextReaderService* is, first analyzed as described in the Input handling phase, and then sent as input for the *TextAnalyzer* class. Next, the *TextAnalyzer* communicates with the *CoreNLPWrapper* sending the input (i.e., process description) which is then transferred via HTTP request to the CoreNLP Server along with the required annotators (2.2.2) through a JSON data format. Internally, the CoreNLP Server triggers its logic for handling the request and providing an output based on the specified annotators and text. Finally, the output is returned via HTTP response to the *CoreNLP-Wrapper*, which creates *StanfordSentence* class objects to encompass the constituency parse, dependency parse and the raw sentence.

Following, each *StanfordSentence* object is sent to the *SentenceAnalyzer* executing methods to extract information from the sentence based on the parses obtained from the CoreNLP pipeline. During this iteration, several interactions between the *SentenceAnalyzer* and the *SentenceElementsBuilder* occur, varying according to the execution flow of the methods. Once every sentence is analyzed, the *TextAnalyzer*'s methods are responsible joining the information of multiple sentences, resolving missing references, completing the *WorldModel* instance and returning it to the *TextReaderService*.

Afterwards, the *TextReaderService* triggers the creation of the objects that represent process elements. This process is performed by the *ProcessElementsBuilder* class

Figure 5.3: Sequence diagram of the server component



Source: The Authors

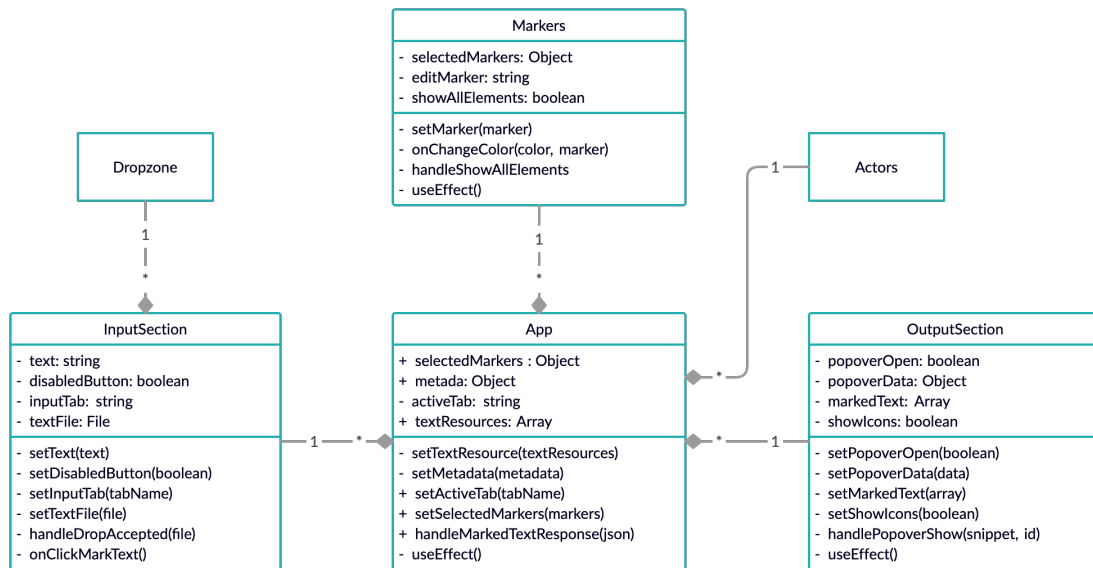
based on information gathered from the WorldModel instance. A ProcessModel class instance is returned encapsulating all the identified process elements. Finally, the ProcessElementsIdentifier class uses the information contained in both data-centered class instances for generating a metadata that links a process element with its respective location in the sentence. The output of this class is returned in a JSON format to the TextReaderService, which is responsible for sending the response back to the client component.

## 5.4 Client component

In comparison with the server component, the client component has a wider range of possibilities of implementations that can still follow the approach proposed in this study. The client component represents the whole Text Annotation phase, which implicates that it must visually represent annotated process descriptions, through colors and symbols and present to end-users. In terms of the approach, the specific techniques and resources are flexible for different choices of implementations, meaning that it lies on the prototype's implementation the responsibility of deciding what specifically is used to annotate the process description.

In the context of the developed prototype, the chosen web architecture, programming language and library enabled a straightforward usage of colors and icons. *ReactJS* provides two main advantages which are particularly suitable for the implementation of the Text Annotation phase: a component-based software engineering (CBSE); and an ef-

Figure 5.4: Class diagram of the client component



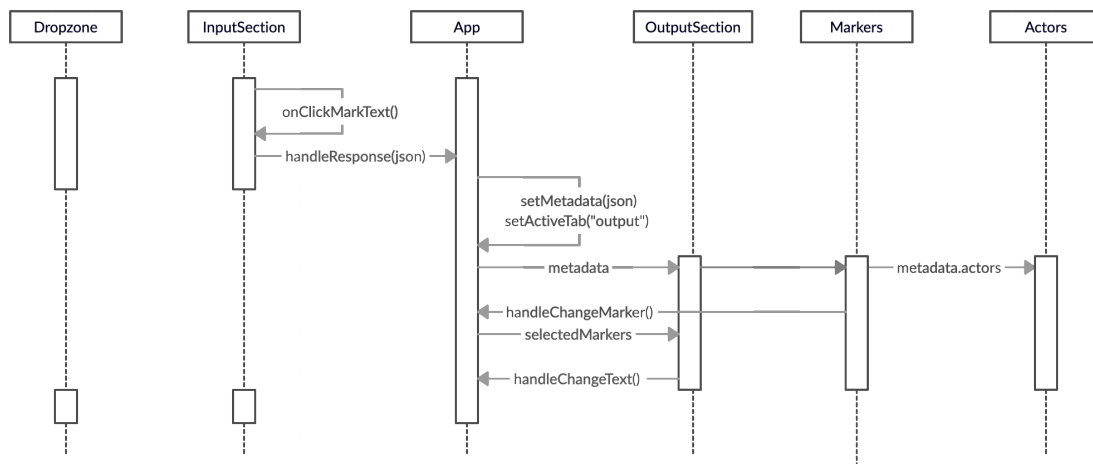
Source: The Authors

fortless integration with Cascading Style Sheets (CSS) and external components (e.g., icon components). Both aspects facilitate an extensible implementation. As per Hovland et al. (2003), CBSE allows the development of independent pieces of functionality as entities that can be composed. Additionally, when defined with clear interfaces, it promotes reusability, as is the case with third-party external components.

Figure 5.4 shows the class diagram of the client component. As per detailed in Fedosejev (2015), the *App* component is the entry point of a ReactJS application. Specifically, this component acts as an instantiator of smaller modular components, splits and orchestrates the application state as: input state or output state. The first is handled by the *InputSection* component, which is responsible for supporting the entry of the process descriptions through native HTML elements such as the *TextArea* or by the *Dropzone* external component, enabling the drag-and-drop functionality of text files. Whenever the end-user submits a process description in the *InputSection*, the *App* component triggers a request to the server component (5.3), waits for a successful response and switches the application to the output state.

The output state, on the other hand, is handled by the *OutputSection*, *Markers* and *Actors* components. The first is responsible for annotating the text with colors and icons according to the metadata received through a JSON format from the server component, which is handled according to the approach presented in the Text Annotation phase. As the technique for applying colors for each word in the output text, a CSS attribute is added to the HTML element representing the respective word. As for the icons addition, an

Figure 5.5: Sequence diagram of the client component



Source: The Authors

external library (ICONIFY, 2019) is used to fetch BPMN 2.0 icons as form of a ReactJS component. The second is responsible for displaying a list with the identifiable process elements names combined with user interactive components such as the `ToggleButton` and `ColorButton`. The precise functionality of those components are further detailed in Section 5.5. Finally, the *Actors* component is responsible for displaying a list of the Actors involved in the input process description.

Unlike the server component, the client component supports a number of possible sequence flows, depending on which action the end-user performs on the web application. Figure 5.5 shows a high level sequence diagram representing the main use case of marking a text and changing which markers are selected. At the initial point of the diagram, the `App` component is at the input state, meaning that the `InputSection` and the `Dropzone` components are instantiated. When the `onClickMarkText` method is triggered and the request is properly handled by the server component, the `handleResponse` method is called from the `InputSection` in order to store the received metadata payload and switch to the output state, at which the `OutputSection`, `Markers` and `Actors` are instantiated. At this point, the `Markers` component handles the interaction with the end-user regarding which process elements are visually annotated on the text. Whenever the end-user switches on or off a marker, the `handleChangeMarker` method is called in order to transfer the information of which markers are currently selected, triggering a change in the visual annotation performed internally in the `OutputSection` component. Finally, the `handleChangeText` method is triggered on the occasion that the end-user decides to change the input process description and restart the sequence.



## 5.5 Demonstration

This section presents a set of features of the prototype developed based on the presented technologies, client and server components. This demonstration aims to display the functionalities available for the end-user of the application, therefore, the content source originates from the client component, running on a web browser. For such purposes, the sequence diagram shown in Figure 5.5 is visually presented with additional use cases, considering the user interactive aspects of the prototype.

As detailed in the previous section, the client component switches between the input and output states, each containing their own sub-components. Figure 5.6 shows the initial input state of the application. At this point, the user has the option to type or paste a process description in the `TextArea` component. Furthermore, in accordance with *Input handling* phase of the presented approach in Section 4.2, the user is also allowed to switch to a file input tab, enabling the selection and upload of a text file, through the `Dropzone` component.

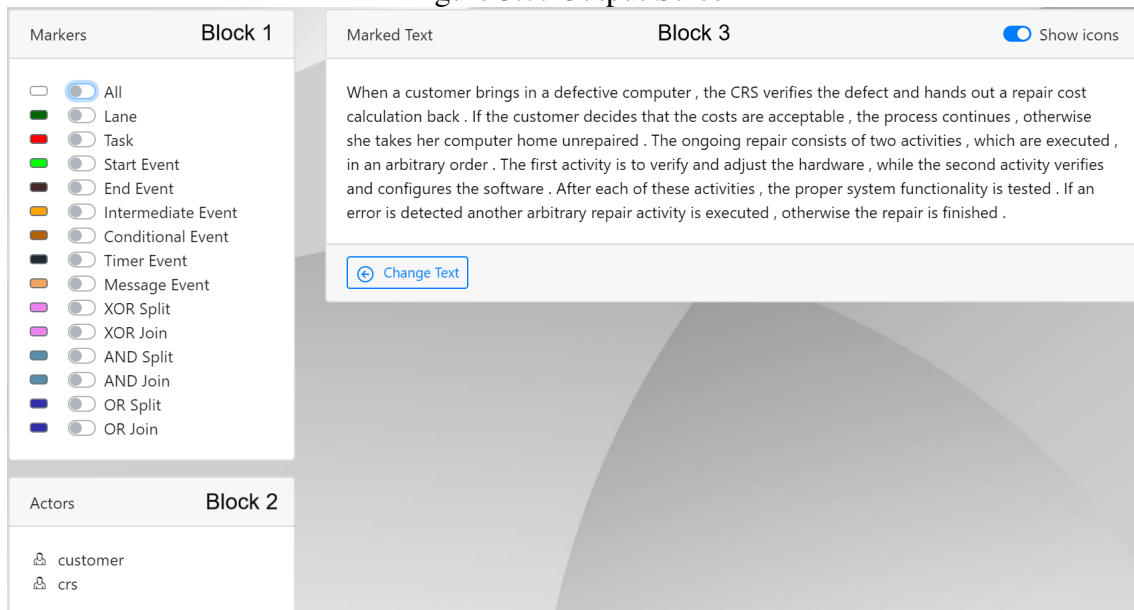
As soon as the *Mark Text* button is pressed, the process description is submitted via HTTP request to the server component. Once the sequence flow of the server component is finished, the client component receives the HTTP response and switches to the output state. The response contains the JSON metadata linking every identified process element with its respective location in the sentence which is handled according to the *Text annotation* phase of the approach. At this point, the user is redirected to the output screen, containing different sorts of information. In Figure 5.7, the output screen is divided in

Figure 5.6: Input text

The screenshot shows a web interface for inputting text. At the top, there is a header bar with the text "Write a process description or upload a text file". Below this, there are two tabs: "Text Input" and "File Input". The "Text Input" tab is active, and it contains a large, empty text area. At the bottom of the interface, there is a blue button labeled "Mark Text".

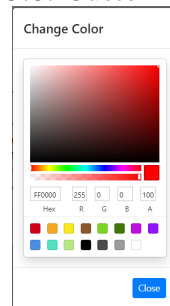
Source: The Authors

Figure 5.7: Output Screen



Source: The Authors

Figure 5.8: Customize color



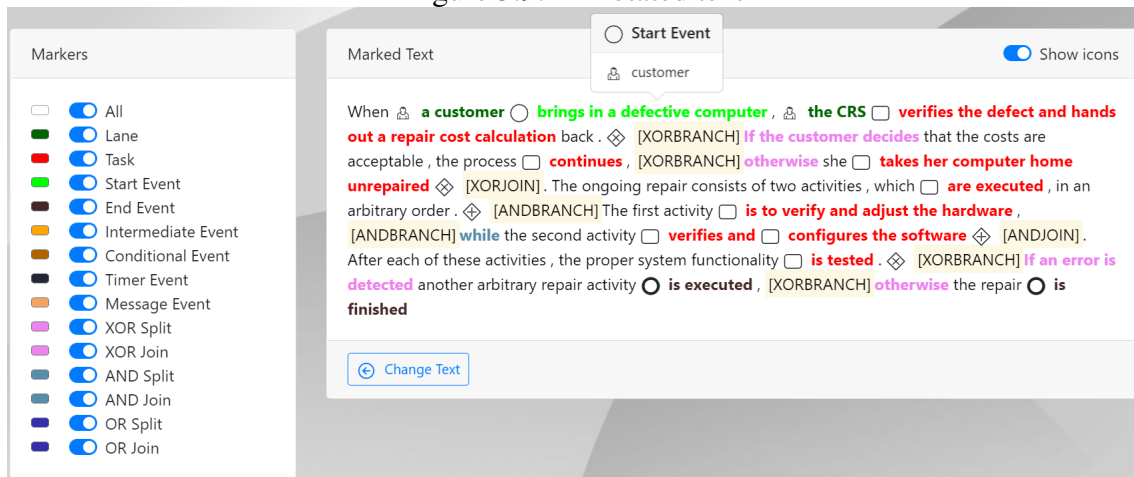
Source: The Authors

three main blocks of information.

*Block 1* contains an interactive set of toggle buttons representing each type of BPMN 2.0 element which is detectable by the approach. Toggling a button in this block triggers the text annotation in *block 3*, resulting in a change of color in the sequences of words that represent the respective BPMN 2.0 element. Furthermore, the process element's icon (COMMISSION et al., 2013) is also displayed before the first word of each sequence. In addition, if the corresponding element is a split gateway, every identified branch on the text will be preceded with a highlighted text indicating the gateway type. Analogously, every identified join gateway is succeeded by a corresponding text indicator.

One of the extended use cases of the client component is the possibility of modifying the color which represents an element type. Clicking on the color button next to the toggle button pops up a dialog, allowing the user to choose the color that represents the respective process element type, as illustrated in Figure 5.8. This functionality aims to enhance the interactivity with the user, allowing a personal customization of the annotated

Figure 5.9: Annotated text



Source: The Authors <sup>1</sup>

text, which enhances the operability of the application. For instance, it is particularly helpful for users with color vision deficiency. *Block 2*, on the other hand, contains a list of every actor extracted from the process description. Such information may be useful to define the lanes and pools representing the process participants in a process model.

Following, *block 3* displays the entry text and all the selected visual features. Whenever the user hovers through an annotated sequence of words (i.e., a process element), its corresponding actor is displayed. This functionality is advantageous, for instance, to assign the process element to its respective lane, during the creation of a process model. In addition, a toggle button allows the user to show or hide icons, with the purpose of avoiding a cognitive overload of graphical elements on the screen. Moreover, the *Change Text* button triggers the application to return to the input state, where a different process description can be entered. Finally, Figure 5.9 shows output screen with every process element selected and a mouse hover through a sequence of words representing a Start Event.

<sup>1</sup>The prototype developed is available in: <<https://github.com/Leosr6/TextAnnotator>>

## 6 EVALUATION OF THE APPROACH

The evaluation of the approach presented in this study was performed by two separated experiments. This chapter details the first experiment, which is based on a user-centered survey, through the means of applying a questionnaire in which participants were presented with the original process description and afterwards, with the annotated process description.

In order to evaluate the approach with no regional or language barriers, two on-line questionnaires were created, one version with questions in Portuguese and another version with questions in English. Both versions were published in social media platforms and were available for any participants from the 4<sup>th</sup> of October of 2020 until the 13<sup>th</sup> of October of 2020. Despite the language, every question had the same meaning and purpose. In the period of 10 days, a total of 32 answers were obtained, combining both versions of the questionnaire.

### 6.1 Questionnaire structure

The questionnaire was created using the widely known Google Forms<sup>1</sup> application, enabling a straightforward and flexible construction of questionnaires. Its overall structure is composed of four sections. The first section, aims to gather information regarding the experience of the participants. Figure 6.1 shows the corresponding section, for which the questions are:

1. *Occupation*: aims to identify the current professional occupation of the participant. The possible options were: “student”, “professor/lecturer”, “software development-related job”, “business process-related job” and “others”.
2. *Education level*: aims to identify the education level of the participant. The possible options were: “highschool completed”, “bachelor’s degree or equivalent in progress”, “bachelor’s degree or equivalent completed”, “master’s degree in progress”, “master’s degree completed”, “PhD degree in progress”, “PhD degree completed” and “others”.
3. *How long have you known about business process modeling*: aims to identify the time of experience of the participant with business process modeling. The possible

---

<sup>1</sup><https://www.google.com/forms/about/> - last access on 18<sup>th</sup> of October of 2020

Figure 6.1: First section of the questionnaire

**About you**

In this section, we will ask questions related to your education, professional experience and BPM knowledge.

**Occupation \***

- Student
- Professor/lecturer
- Software development-related job
- Business process-related job (analyst, engineer, etc.)
- Outro: \_\_\_\_\_

**Education level \***

- Highschool completed
- Bachelor's degree or equivalent in progress
- Bachelor's degree or equivalent completed
- Master's degree in progress
- Master's degree completed
- PhD degree in progress
- PhD degree completed
- Outro: \_\_\_\_\_

**How long have you known about business process modeling? \***

- I have no knowledge about business process modeling
- Less than 1 year
- Between 1 and 2 years
- Between 2 and 3 years
- More than 3 years

**How long have you known about the Business Process Model and Notation (BPMN)? \***

- I have no knowledge about BPMN
- Less than 1 year
- Between 1 and 2 years
- Between 2 and 3 years
- More than 3 years

**What is your level of knowledge about the English grammar? \***

1      2      3      4      5

Very little knowledge                        High domain

Source: The Authors

options were: “no knowledge about business process modeling”, “less than 1 year”, “between 1 and 2 years”, “between 3 and 4 years” and “more than 3 years”.

4. *How long have you known about the Business Process Model and Notation (BPMN):* aims to identify the time of experience of the participant with the BPMN, considering that the approach uses BPMN 2.0 elements and icons to annotate the description. The possible options were: “no knowledge about BPMN”, “less than 1 year”, “between 1 and 2 years”, “between 3 and 4 years” and “more than 3 years”.
5. *What is your level of knowledge about the English grammar:* aims to identify the level of knowledge of the participant about the English grammar. The answer is based on the Likert scale (LIKERT, 1932), composed of five points, varying from “very little knowledge” up until “high domain”.

The following section of questionnaire presents a process description to the participant, divided in three sentences. Then, for each sentence, the participant is asked to select the amount of each process element type present in sentence. As illustrated by Figure 6.2, the possible options for each question are the process elements identifiable by the approach (fig. 2.1) varying from an amount of 0 up until more than 4. Every process element in the list required a selection. The purpose of this section is to check how effectively participants can identify process elements in a raw process description.

Figure 6.2: First sentence in the second section of the questionnaire

**First part**  
 When a customer brings in a defective computer, the CRS verifies the defect and hands out a repair cost calculation back. If the customer decides that the costs are acceptable, the process continues, otherwise, she takes her computer home unrepaired.

How many process elements do you consider to exist in this part? \*

	0	1	2	3	4	More than 4
Lane	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Task	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Start Event	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
End Event	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Conditional Event	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Timer Event	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Message Event	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
XOR Split	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
XOR Join	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
AND Split	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
AND Join	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
OR Split	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
OR Join	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Others	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Source: The Authors

Afterwards, the third section of questionnaire begins presenting the color legend for each type of process element, as shown in Block 1 of Figure 5.7. This section explains that the process description used in the previous section had its process elements annotated by the prototype according to the color legend and, finally, displays the completely annotated process description, as shown in Figure 5.9. Directly after the explanatory phase, the participant is asked questions regarding the annotated text. Unlike the previous section of questionnaire, these questions aim to implicitly identify whether the participant improved the comprehension about the business process based on a process description with annotated process elements. Figure 6.3 shows the third section of this questionnaire, for which the questions are:

1. *On a scale of ease, how do you consider understanding the annotated text:* this

Figure 6.3: Third section of the questionnaire

Annotation of elements in process descriptions

In this section, we will present a visual annotation of the process elements contained in the process description shown in the previous section.

The description of the referred process had its process elements annotated with colors according to the legend shown below:

- All
- Lane
- Task
- Start Event
- End Event
- Intermediate Event
- Conditional Event
- Timer Event
- Message Event
- XOR Split
- XOR Join
- AND Split
- AND Join
- OR Split
- OR Join

**More about the annotation**  
Besides the colors represented above, the sentences in the text below were annotated with symbols according to BPMN.

Furthermore, each branch of a given "Split" is preceded by one of the following text annotations: "[XORBRANCH]", "[ANDBRANCH]" or "[ORBRANCH]".

The "Joins", in turn, are followed by a text annotation of one of the following types: "[XORJOIN]", "[ANJOIN]" or "[ORJOIN]".

According to the annotated text below, answer the following questions:

When a customer brings in a defective computer, the CRS verifies the defect and hands out a repair cost calculation back. [XORBRANCH] If the customer decides that the costs are acceptable, the process continues. [XORBRANCH] otherwise she takes her computer home unrepaired. [XORJOIN]. The ongoing repair consists of two activities, which are executed, in an arbitrary order. [ANDBRANCH] The first activity is to verify and adjust the hardware, [ANDBRANCH] while the second activity verifies and configures the software. [ANDJOIN]. After each of these activities, the proper system functionality is tested. [XORBRANCH] If an error is detected another arbitrary repair activity is executed. [XORBRANCH] otherwise the repair is finished.

How many "tasks" are annotated?

There are no "tasks" annotated

4

6

8

10

Outro: \_\_\_\_\_

On a scale of ease, how do you consider understanding the above text? \*

1   2   3   4   5

It was extremely difficult                  It was extremely easy

Which combination will never occur during the process execution? \*

'verifies the defect and hands out a repair calculation' & 'configures the software'

'the process continues' & 'she takes her computer home unrepaired'

'system functionality is tested' & 'the repair is finished'

'another arbitrary repair activity is executed' & 'the repair is finished'

'verify and adjust the hardware' & 'configures the software'

Outro: \_\_\_\_\_

Who are the actors of the process? \*

There are no actors in this process

'Customer'

'Customer' & 'CRS'

'Customer' & 'Computer'

'Computer'

Outro: \_\_\_\_\_

Which activities may occur in parallel during the process execution? \*

'another arbitrary repair activity is executed' & 'the repair is finished'

'verifies the defect and hands out a repair calculation' & 'configures the software'

'verify and adjust the hardware' & 'configures the software'

'system functionality is tested' & 'the repair is finished'

'the process continues' & 'she takes her computer home unrepaired'

Outro: \_\_\_\_\_

Source: The Authors

question aims to identify if the participant has any issues in reading the text, such as: cognitive overload, recognition of the BPMN 2.0 symbols, etc. The answer is based on the Likert scale, varying from “extremely difficult” to “extremely easy”.

2. *Who are the actors of the process*: aims to identify who are the actors of the business process according to the participant of the questionnaire. The possible options were: “there are no actors in the process”, “customer”, “customer & CRS”, “customer & computer”, “computer” and “others”.
3. *How many tasks are annotated*: a question directed towards the understanding of the annotated in the process description, by asking the amount of *task* elements annotated. The possible options were: “there are no tasks annotated”, “4”, “6”, “8”, “10” and “others”.
4. *Which combination will never occur during the process execution*: a multiple choice question which aims to identify if the participant can define tasks in branches belonging to the same exclusive decision point. The possible options were: “verifies the defect and hands out a repair calculation & configures the software”, “the process continues & she takes her computer home unrepaired”, “system functionality is tested & the repair is finished”, “another arbitrary repair activity is executed & the repair is finished”, “verify and adjust the hardware & configures the software” and “others”.
5. *Which activities may occur in parallel during the process execution*: this question aims to identify if the participant can define tasks in branches belonging to the same parallel execution. The possible options were: “another arbitrary repair activity is executed & the repair is finished”, “verifies the defect and hands out a repair calculation & configures the software”, “verify and adjust the hardware & configures the software”, “system functionality is tested & the repair is finished”, “the process continues & she takes her computer home unrepaired” and “others”.

Finally, the last section of the questionnaire contains feedback focused questions, based on the participant’s point of view regarding the overall annotated text. Unlike the previous sections, this particular section contains two qualitative questions which are optional for the completion of the questionnaire. The purpose of this section is to evaluate effectiveness and discover points of improvement for the annotation approach. As illustrated by Figure 6.4, this section begins by recalling the annotated process description presented in the previous section and continues with the following questions:



Figure 6.4: Fourth section of the questionnaire

Feedback about the annotation in the process description

In this section, we will ask your opinion regarding the annotated text in the previous section.

Recall the annotated process description of the previous section and answer the questions below:

When a customer brings in a defective computer, the CRS verifies the defect and hands out a repair cost calculation back. If the customer decides that the costs are acceptable, the process continues, otherwise she takes her computer home unrepaired. The ongoing repair consists of two activities, which are executed, in an arbitrary order. The first activity is to verify and adjust the hardware, while the second activity verifies and configures the software. After each of these activities, the proper system functionality is tested. If an error is detected another arbitrary repair activity is executed, otherwise the repair is finished.

In a scale of correctness, how much do you agree with the annotation in the process description? \*

1 2 3 4 5

Completely disagree      Completely agree

How much would this annotated process description help you to build the process model in comparison with the original process description? \*

1 2 3 4 5

Not helpful at all      Extremely helpful

Do you have any criticism, comments or suggestions regarding this annotation in the process description?

If so, we ask you to answer the following question as well.

Sua resposta \_\_\_\_\_

Process description used in this questionnaire

When a customer brings in a defective computer, the CRS verifies the defect and hands out a repair cost calculation back. If the customer decides that the costs are acceptable, the process continues, otherwise, she takes her computer home unrepaired. The ongoing repair consists of two activities, which are executed, in an arbitrary order. The first activity is to verify and adjust the hardware, while the second activity verifies and configures the software. After each of these activities, the proper system functionality is tested. If an error is detected another arbitrary repair activity is executed, otherwise, the repair is finished.

What would an ideal annotated process description look like in your opinion?

Annotate the process description (there is a copy above) using a text editor (Word, Google docs, etc.). Use the color legend indicated in previous section, export it as PDF and upload the file below.

[Adicionar arquivo](#)

Source: The Authors

1. *In a scale of correctness, how much do you agree with the annotation in the process description:* aims to gather a user based validation of the annotation approach on the presented process description. The answer follows the Likert scale, varying from “completely disagree” up until “completely agree”.
2. *How much would this annotated process description help you to build the process model in comparison with the original process description:* a question directed towards one possible use case of the presented approach, which is further demonstrated in Chapter 7. The answer follows the Likert scale, varying from “not helpful at all” up until “extremely helpful”.
3. *Do you have any criticism, comments or suggestions regarding this annotation in the process description:* a qualitative free text based question with the purpose of obtaining feedback about the annotation used throughout the questionnaire.
4. *What would an ideal annotated process description look like in your opinion:* a qualitative question aiming to gather different possible annotations for the process description based on the participant’s opinion. The answer is a file input allowing the submission of a PDF file containing the annotated process description.

## 6.2 Result analysis

Based on the application of the questionnaire, the evaluation of the presented approach was performed by analyzing the results in a quantitative and qualitative manner. As such, a set of four main points were addressed by the questions in the questionnaire in order to achieve the overall evaluation of the approach. The first point is concerned to the *comprehension* of the annotated text in comparison with the original text in terms of readability. A second point relates to the *correctness* of the annotation approach on the presented process description. Following, the third point is regarding the *usefulness* of the annotation for the design of the process model. Finally, the fourth point reflects the *effectiveness* of the approach for the interpretation of the business process.

As the process description presented in the questionnaire is written in English, a minimum level of knowledge about the English grammar is required to understand it. Therefore, submissions with values of 1 and 2 on the Likert scale regarding the English grammar knowledge question could generate noise for evaluating the approach. As such, the submissions were filtered for answers with a value equal or higher than 3 in the mentioned question. As a result, 1 submission was disconsidered and 31 remained.

According to Allen and Seaman (2007), the analysis of Likert scale based data can be performed by collapsing the responses into condensed categories. In that regard, answers in the scale were set as one of three categories: *positive*, *neutral* or *negative* answers. The positive category encompasses answers between the range of 4 and 5 in the scale. As oppositely, the negative category encompasses answers between the range of 1 and 2 in the scale. Last, the neutral category represents the value 3 in the scale.

In particular, answers following the Likert scale can provide straightforward insights, however, an isolated analysis may indicate misleading results. On the other hand, combining answers based on the Likert scale with the experience of the participants, obtained in the first section of the questionnaire, leads to a more granular analysis. In spite of the data combination, however, the results based on Likert scale questions are nevertheless dependent on the participant's evaluation.

On the other hand, the second and third sections of the questionnaire were designed to provide an unbiased insight of the effectiveness of the approach. In essence, combining the results of questions regarding the original process description with results of questions regarding the annotated process description lead to a clear insight of the difference between them in terms of the participant's comprehension of the business process.

Such analysis is particularly useful to address the fourth point (i.e., approach effectiveness) of the evaluation.

### 6.2.1 Comprehension of the annotated description

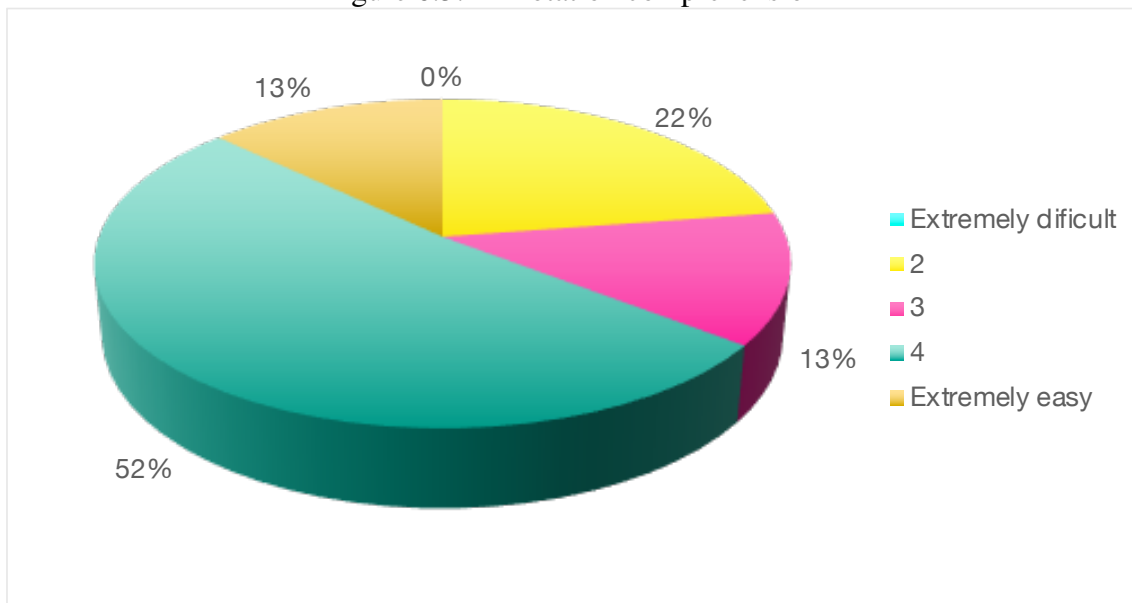
As means to cover the first point of the evaluation, i.e., the comprehension of the annotated text, the analysis is performed based on the first question of the third section of the questionnaire (Figure 6.3). Considering this Likert scale based question, a general indicator can be achieved by plotting the answers of the question in an isolated manner. On the other hand, a more valuable insight is to analyze the comprehension of the annotated process description based on an occupation perspective. Based on this information, it is possible to infer which occupation benefits the most from the annotated process description. Figures 6.5 and 6.6 show the isolated and combined results of the first point of the evaluation, respectively.

As shown in Figure 6.5, the overall annotation comprehension tended towards positive answers, comprising a total of 65% of the submissions. The negative answers comprised a total of 22% of the submissions for which none represented the lowest value in the scale (i.e., “extremely difficult”). Finally, 13% of the submissions were neutral answers. The annotation comprehension based on the participant’s occupation, illustrated in Figure 6.6, is best described by absolute values rather than percentages, as it also shows the amount of participant’s in each occupation. In order to comprehend this result, it is necessary to fallback to a qualitative analysis, based on the free text feedback provided in the last section of the questionnaire.

The comprehension for students fits mostly into the positive category with 8 submissions, however, 2 submissions had neutral answers and none had negative answers. The related improvement feedback contains comments such as: “*modify the background color of the text, instead of the color text*”, “*the symbols complicate the reading*” and “*add parenthesis/brackets/braces surrounding the branch sentences in order to clarify the annotation*”. All of which are possible to achieve following the presented approach, with modifications in the client component of the prototype. Particularly, removing the symbols is already a possibility in the prototype.

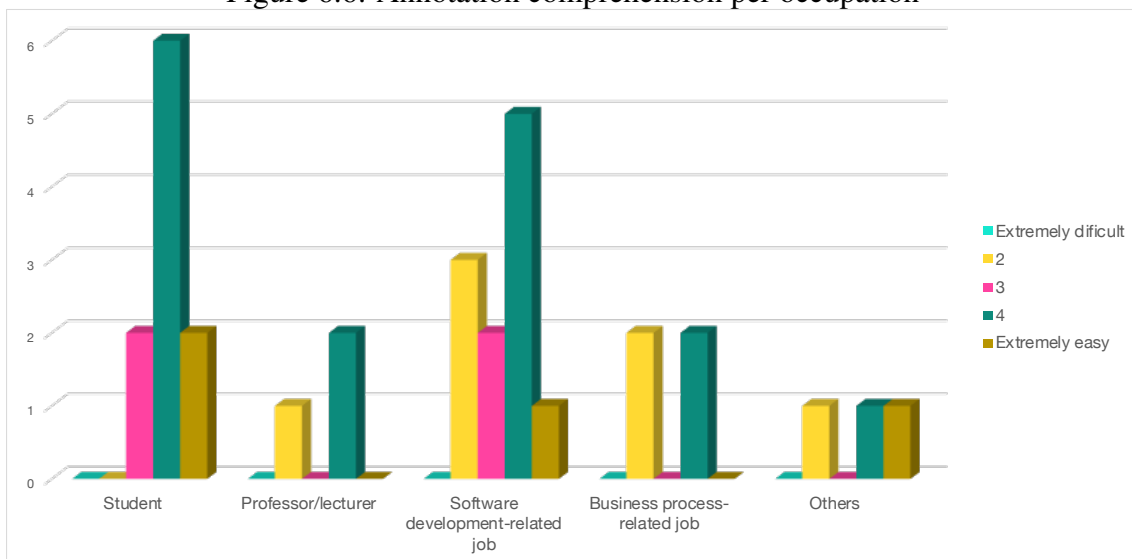
Professors and lecturers represent a minority in the total population of the experiment. While 2 submissions leaned towards positive answers, 1 submission presented a negative answer. The reason behind the difficult comprehension, however, is not stated in

Figure 6.5: Annotation comprehension



Source: The Authors

Figure 6.6: Annotation comprehension per occupation



Source: The Authors

the feedback.

Software development-related jobs, on the other hand, represent the majority of participant's of the experiment. The results for this category are scattered, although overall tending towards positive answers with 6 submissions, negative and neutral answers are present in 3 and 2 submissions, respectively. The related improvement feedback contains comments such as: *“the amount of information at once is a bit illegible”*, *“the representation of gateways is confusing”*, *“visualize gateway branches either with indentation or something similar”*. Although reducing the amount of information is already covered in the prototype, the representation of branches could be a point of improvement.

The business process-related job category comprises of 4 submissions equality distributed as 2 positive and 2 negative answers. Even though no feedback related to the comprehension of the text was provided, one key factor is that this category encompasses multiple occupations, e.g., process analyst, process engineer. As such, either a bigger sample or more granular occupation information is required to explain the results.

Additionally, 3 submissions contained occupations not covered by the preselected categories, being “customer service manager”, “TIC project manager” and “consultant”, with answers of values 4, 5 and 2, respectively. While the positive answers for these occupations had no feedback related to the comprehension of the text, the negative answer stated that “the annotation may have value for an automation tool, but for a human reader the original descriptions seems a better choice”. It is unclear in the commentary, however, the reason why the human reader would benefit more from the original description.

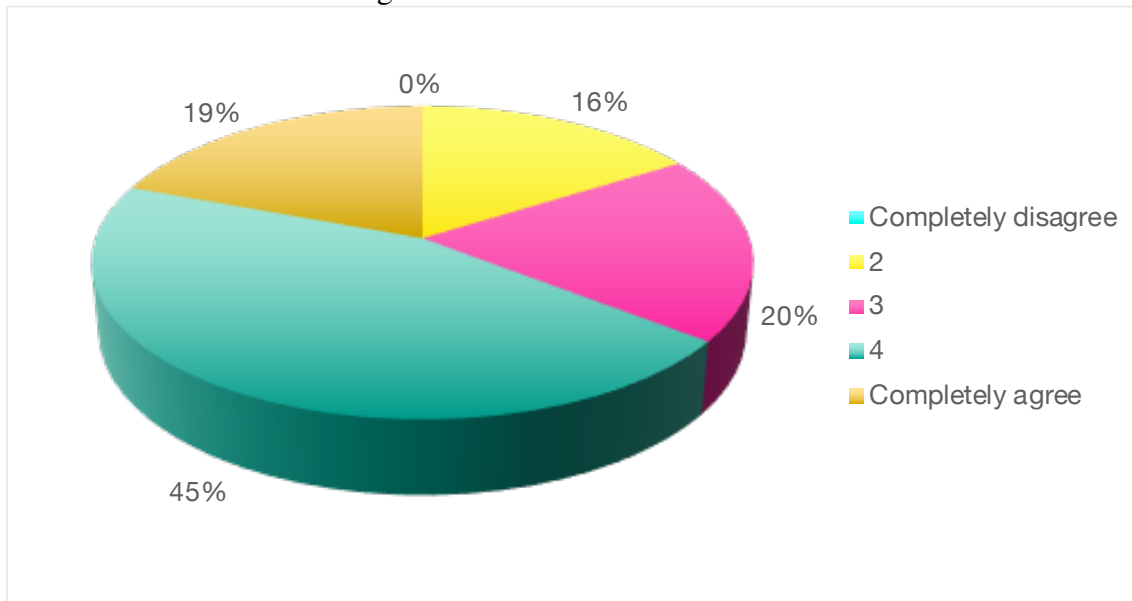
Finally, based on the analysis per occupation, the results point towards a particularly easy comprehension of the annotated description for students, which indicates that the presented approach could also be leveraged, for instance, as a facilitator for teaching purposes. Furthermore, some points of improvements regarding the annotation of gateways and its branches were also presented for improving the prototype.

## 6.2.2 Annotation correctness

The second point of the evaluation, i.e., correctness of the annotation in the process description, is performed based on the first question of the fourth section of the questionnaire (Figure 6.4). Similarly to the previous point, the question is based on the Likert scale, which can be individually analyzed, indicating the general correctness of the annotation. On the other hand, a more valuable analysis is achieved when also considering the participant’s experience with the BPMN. Figures 6.7 and 6.8 show the isolated and combined results of the second point of the evaluation, respectively.

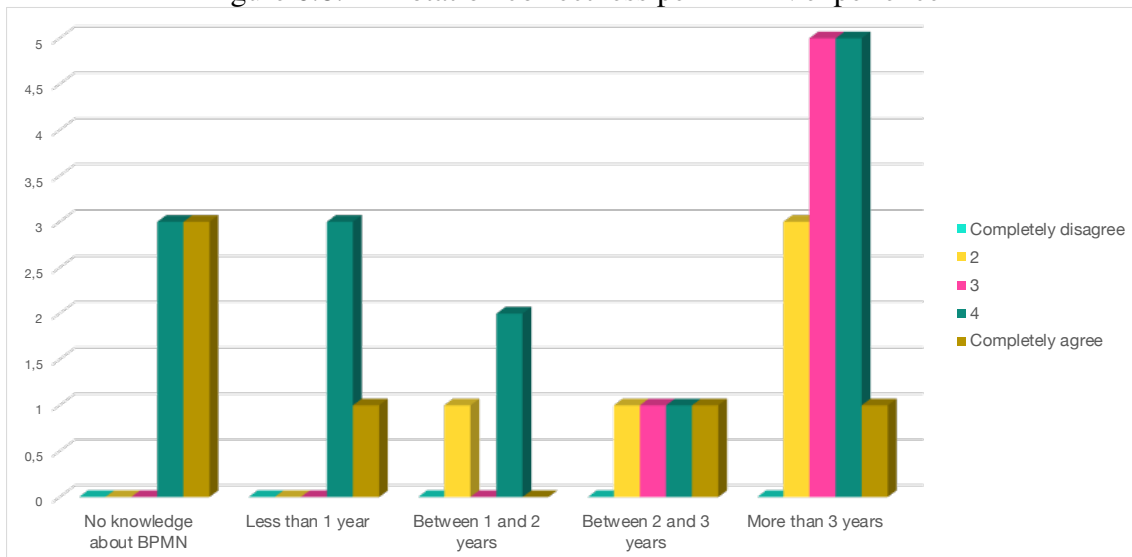
As shown in Figure 6.7, the annotation correctness has also tended towards positive answers, comprising a total of 64% of the submissions. Negative answers comprised a total of 16% of the submissions for which none represented the lowest value in the scale (i.e., “completely disagree”). Finally, neutral answers corresponded to 20% of submissions. The annotation correctness based on the participant’s BPMN experience, illustrated in Figure 6.8, also demonstrates, in absolute values, that experienced BPMN participants represent the largest sample in the population of the experiment. It is assumed that the

Figure 6.7: Annotation correctness



Source: The Authors

Figure 6.8: Annotation correctness per BPMN experience



Source: The Authors

more experienced a participant is with the BPMN, the more qualified he/she is to judge the correctness of the annotation.

Participants with no knowledge and less than 1 year of knowledge about BPMN add to a total of 10 submissions, all which tending towards positive answers. Although these answers are presumably not the most qualified for evaluating the correctness of the annotation, the related feedback contains comments with some valuable insights. The first topic noted is regarding the separation of the tasks “*verifies and configures the software*”, while the tasks “*is to verify and adjust the hardware*” are annotated as a single task. This is a point of improvement for the Elements extraction phase of the approach, in which the

statistical based model used in the Stanford CoreNLP identifies the “is” as the main verb of the sentence, rather than “verify” and “adjust”.

A second topic noted concerns the task “*takes her computer home unrepaired*”, which is followed by a join gateway. In terms of business process execution, however, the process would also finish at this point, therefore, additional logic is required to detect an end event at this point. Last, the third topic raised the discussion of whether the annotation of the tasks “continues” and “are executed” may cause confusion as to which precise action is to be performed during the execution of the process.

Participants which have between 1 and 2 years and between 2 and 3 years of experience with the BPMN can provide a more qualified evaluation of the annotation correctness. The submissions lean towards positive answers, adding to the amount of 4 combining both categories of experience. Negative answers add to the amount of 2, while a single neutral answer was submitted. For both categories, no comment related to the correctness of the annotation was provided.

Presumably being the most qualified sample of the experiment for the evaluation of this point, submissions of participants having more than 3 years of experience with the BPMN also tended towards positive answers, adding to the amount of 6 submissions. Unlike in the previously analyzed categories, the neutral answers represent a considerable amount of 5 submissions. Negative answers, on the other hand, represent the lowest amount in this category, with 3 submissions.

The related feedback contains comments regarding the second and third topics raised in the analysis of the previous categories, i.e., the lack of the end event after the task “*takes her computer home unrepaired*” and the annotation of tasks which signal the continuity of the process. An additional point was raised in two comments, concerning the annotated task “verifies the defect and hands out a repair cost calculation”. Both participants stated that this task would be best represented as two separate tasks, due to the complexity of each action. One of the comments mentions that this would facilitate to identify sub-processes for each of these tasks.

Although some improvement points regarding the annotation of tasks were identified, the analysis of the results indicate that the correctness of the annotated text is positive for every category of BPMN experience. Furthermore, the inconsistencies in the annotation are focused on splitting aggregated tasks, rather than incorrect type of BPMN elements being annotated.

### 6.2.3 Usefulness for modeling purposes

In order to cover the third point of the evaluation, i.e., usefulness of the annotated text as an artefact for modeling purposes, the analysis is performed based on the second question of the fourth section of the questionnaire (Figure 6.4). Following the same pattern of the already discussed points, the answers to this question are analyzed individually and combined with the participant's experience with process modeling. Particularly, this point is also further explored as an use case, in Chapter 7. Figures 6.9 and 6.10 show the isolated and combined results of the third point of the evaluation, respectively.

As illustrated in Figure 6.9, the participant's evaluation of the usefulness of the annotation approach for a process model design use case is strongly inclined towards positive answers, comprising a total of 88%. Unlike the two previous points, the majority of answers correspond to the highest value in the scale (i.e., "extremely helpful"), even though a single submission contained the lowest value in the scale (i.e., "not helpful at all"). The annotation usefulness for model design based on the participant's experience with process modeling is shown with absolute values in Figure 6.10, which also demonstrates that the largest sample in the population of the experiment have more than 3 years of experience with process modeling. Similarly to the previous point, it is assumed that the more experienced a participant is with process modeling, the more qualified he/she is to judge the usefulness of the approach for such purpose.

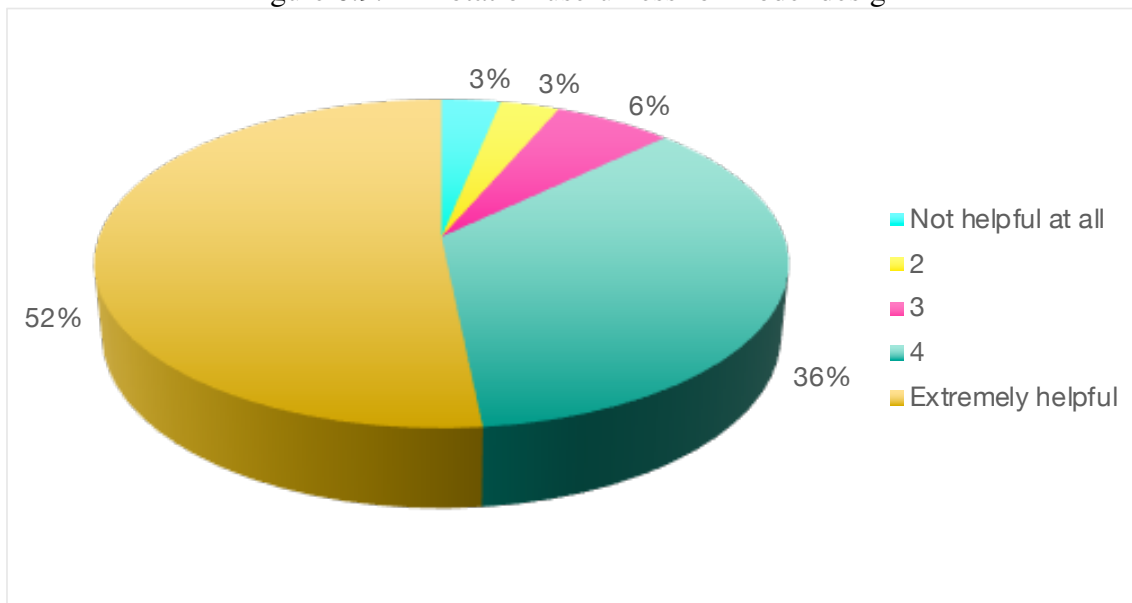
Participants which have no knowledge about business process modeling comprise a small sample of the population of the experiment, which is a positive factor for the evaluation of this point. The amount of participants with no experience and less than 1 year of experience in process modeling adds to a total of 9 submissions, all of which fitting in the category of positive answers. In particular, 7 of them represent the highest value in the scale. For both categories, no comment related to the modeling usefulness was provided.

The submissions of participants which have between 1 and 2 years and between 2 and 3 years of experience with process modeling also lean towards positive answers, adding to the amount of 5 combining both categories of experience. Nevertheless, negative and neutral answers were accounted, at 1 submission each. For both categories, no comment related to the modeling usefulness was provided.

A second positive factor for the analysis is the fact that the most experienced participants with process modeling comprise the biggest sample of the experiment. As such,

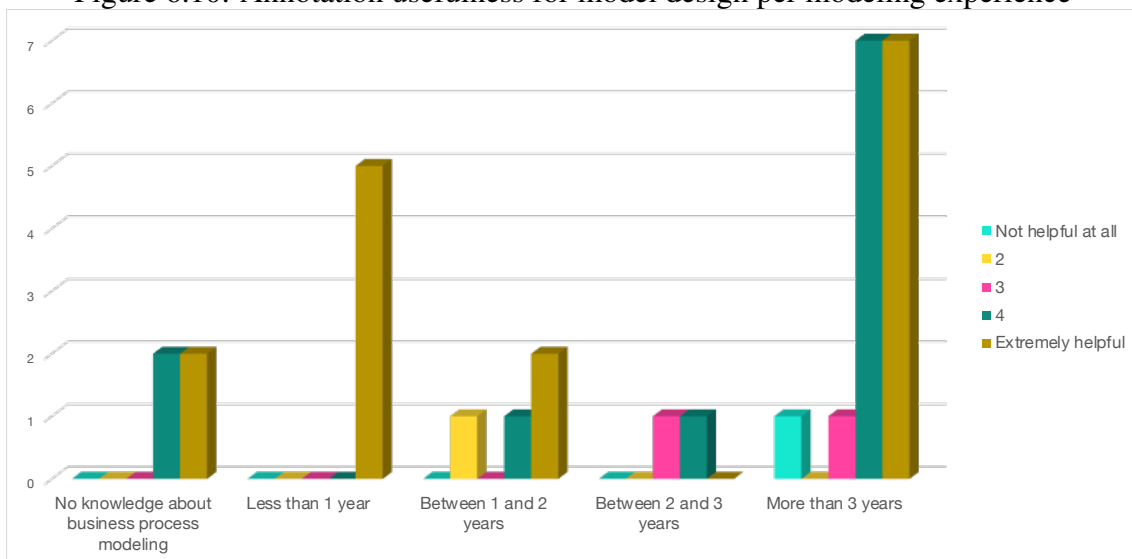


Figure 6.9: Annotation usefulness for model design



Source: The Authors

Figure 6.10: Annotation usefulness for model design per modeling experience



Source: The Authors

the participant's evaluation of this point is presumably more reliable and qualified. The submissions in this category are strongly inclined towards positive answers, adding up to a total of 14. Additionally, only a single neutral answer was accounted, for which the related feedback indicated that providing a distinction between lanes and pools would be important, as it is a recurrent question to be addressed during the process modeling. Also, only a single negative answer was accounted, for which the related feedback was already discussed in the comprehension of the annotated description, for which the participant argued that the original description was a better fit for human readers.

Several feedback contained positive comments regarding the usefulness of the ap-

proach for process modeling, such as: “the annotation would help a lot in the identification of the process steps”, “the annotation practically performs the modeling by it self”. Not only the consolidate results, but also the most qualified category of participants strongly indicate that the annotation approach is helpful for such use case. Finally, the already discussed indentation of gateways could also be a point of improvement to assist in the process modeling.

#### **6.2.4 Effectiveness of the approach**

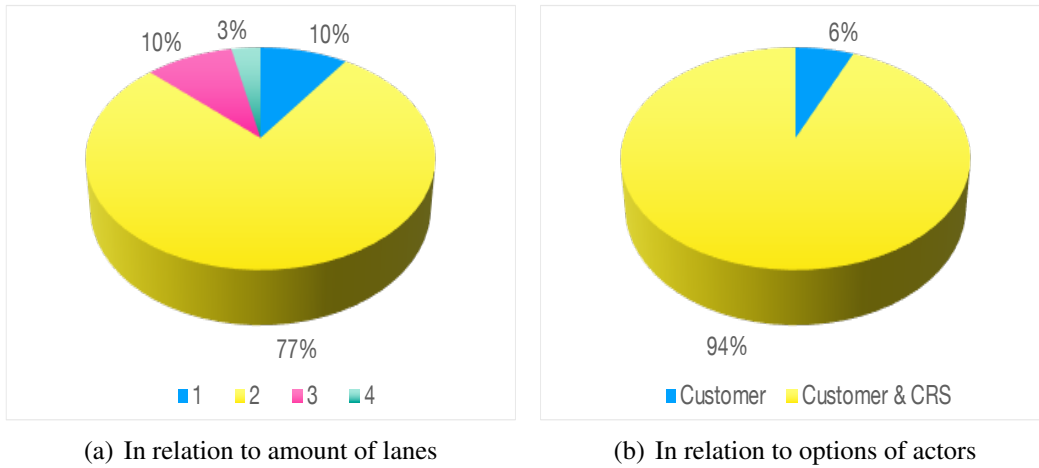
The fourth point of the evaluation, i.e., effectiveness of the approach, is analyzed based on the answers obtained in the third section (Figure 6.3) of the questionnaire, in comparison with the answers obtained in the second section (Figure 6.2). Unlike the previous points, this analysis indicate results which are not based on the participant’s judgement, as such, it is possible to achieve an unbiased view of the effectiveness of the approach. Specifically, the term effectiveness stands for the capability of the approach to be similar - or better - compared to a human’s interpretation of the process description.

In the second question of the third section, the participants were asked to select the actors of the process, based on the annotated text, between a number of options. As explained in Section 2.1, in the BPMN 2.0, actors are later generally represented as lanes. Therefore, in order to evaluate the identification and annotation of the actors in the process description, a comparison is performed with the amount of lanes previously answered in the second section of the questionnaire. Figure 6.11 shows the percentage of participants in relation to the number of lanes selected and in relation to the options of actors in the third section.

As illustrated, the first image shows that 77% of the participants selected an amount of 2 lanes in the process description in the second section. In comparison, the second image shows that 94% of the participants selected the option “Customer & CRS” as the annotated actors in the process description. Considering that none of the participants answered with the “Others” option, indicating different actors out of the available options, it is possible to infer that the annotation of the actors improved the interpretation of some participants concerning the actors in the process.

Unlike the previous question, the third question of the section can be directly compared with the answers in the second question, as both lead to quantitative answers. As such, it is possible to perform a correlation between the amount of tasks selected in both

Figure 6.11: Actors annotation comparison



Source: The Authors

sections for each submission, as demonstrated in Figure 6.12. However, due to the inconsistencies of the annotation of tasks discussed in the evaluation of the *annotation correctness* point, a binary comparison would be misleading, as participants may have considered some tasks separately or aggregated.

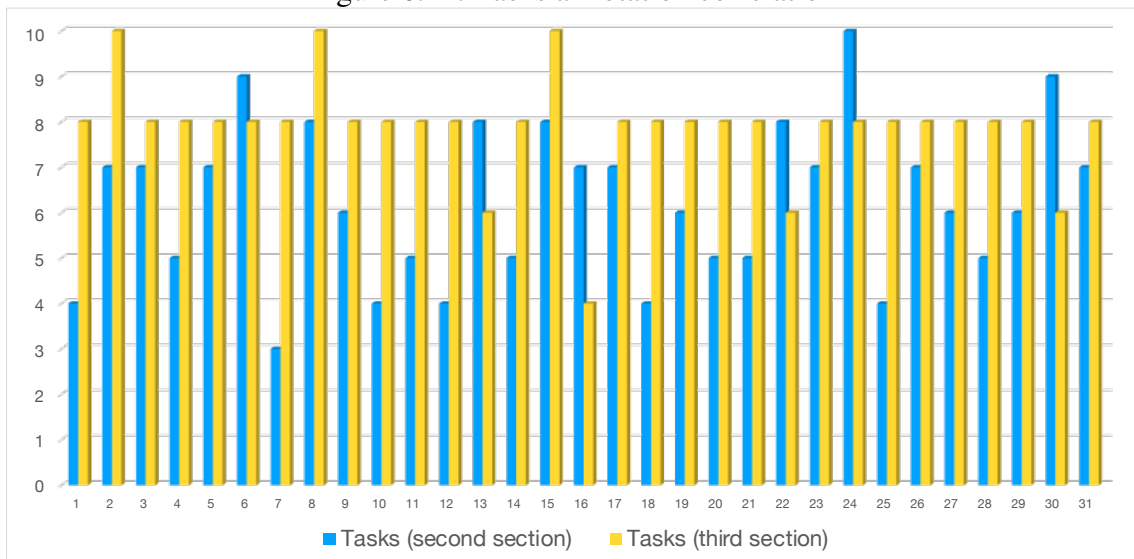
In order to overcome this issue and still have value from the correlation of the amount of tasks selected, the *pooled standard deviation* is calculated, combining the standard deviation of the amount of tasks in both sections of every submission. The pooled standard deviation is calculated as per equation 6.1, where:  $k$  is the number of submissions (i.e., 31);  $n$  is the number of questions considered in the standard deviation (i.e., 2), which are the selection of tasks in the second and third section; and  $s^2$  represents the variance of the respective submission  $i$ .

$$\sigma_p = \sqrt{\frac{\sum_{i=1}^k (n_i - 1) * s_i^2}{\sum_{i=1}^k (n_i - 1)}} \quad (6.1)$$

$$CV = \frac{\sigma_p}{\mu} \quad (6.2)$$

As a result, the overall standard deviation between the amount of tasks selected in the second and third section of the questionnaire is of value 1,35. In order to further analyze this value, a second metric is required, calculating the *coefficient of variation* (CV), which is defined as the ratio of the standard deviation in relation to the mean. In this case, it is the ratio between the pooled standard deviation and the mean of the amount of tasks of both questions in every submission. The CV is calculated as per equation 6.2 and results in the value of 0,19. As rule of the thumb, values lower than 1 are considered to be of low variance, which is an indicator that the difference between the amount of

Figure 6.12: Tasks annotation correlation



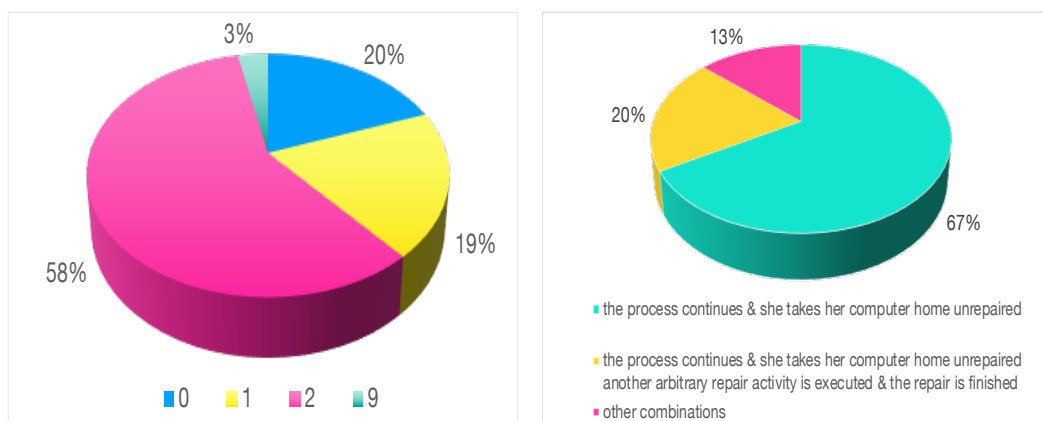
Source: The Authors

tasks selected in both sections is low, despite the already mentioned inconsistencies.

The fourth question of the section checked whether the participants could identify the tasks which occur in different branches of a same exclusive decision point. In order to evaluate the identification and annotation of XOR splits, a comparison is performed between the answers for this question and the amount of XOR splits previously answered in the second section of the questionnaire. As this is a multiple choice question, unlike the previous questions, the analysis is based on the combinations of answers submitted. Figure 6.13 shows the percentage of participants in relation to the number of XOR splits selected and in relation to the combinations of exclusive tasks in the third section.

As illustrated, the first image shows that 58% of the participants selected an amount of 2 XOR splits in the process description in the second section. However, as the second

Figure 6.13: Exclusive decisions annotation comparison



(a) In relation to amount of XOR splits

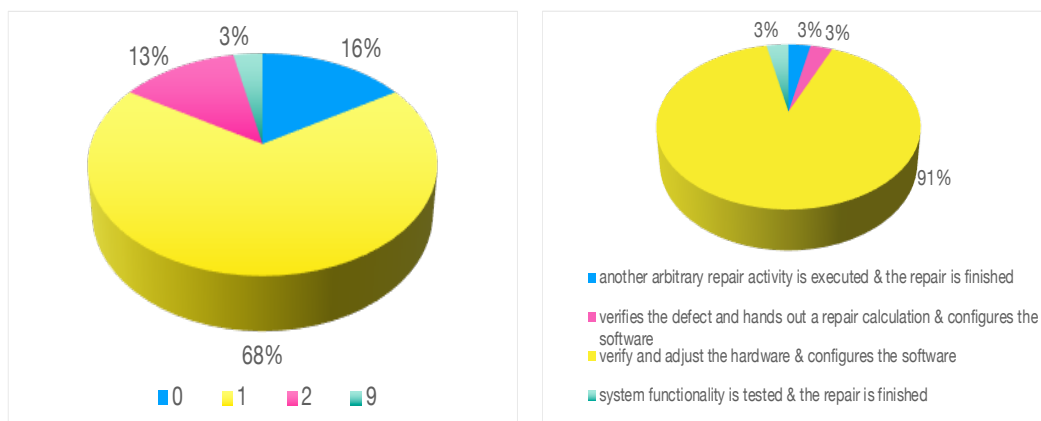
(b) In relation to combinations of exclusive tasks

Source: The Authors

image shows, the majority of participants, representing 67%, selected a single annotated exclusive decision in the third section, whereas 20% selected the set of both XOR splits annotated in the process description. The expectation was that a higher percentage would be able to identify both XOR gateways, specially considering that the majority of participants selected 2 XOR splits in the second section. One possible assumption behind this is that the question is not explicitly stating the more than one option can be selected. On the other hand, it was not possible to confirm this assumption, therefore, the effectiveness of the annotation of exclusive decision points remains uncertain. Finally, different combinations which were selected by a single participant were joined in a category defined as “other combinations” and sum up to a total of 13%.

The last question of the section checked whether the participants could identify the tasks which occur in different branches of a same parallel execution. In order to evaluate the identification and annotation of AND splits, a comparison is performed between the answers for this question and the amount of AND splits previously selected in the second section of the questionnaire. Figure 6.14 shows the percentage of participants in relation to the number of AND splits selected in relation to the options of parallel tasks in the third section.

Figure 6.14: Parallel executions annotation comparison



(a) In relation to amount of AND splits

(b) In relation to combinations of parallel tasks

Source: The Authors

As illustrated, the first image shows that 68% of the participants selected an amount of 1 AND split in the in process description in the second section. In comparison, the second image shows that 91% of the participants selected the option “verify and just the hardware & configures the software” as the annotated parallel tasks in the process description. Similar to the case of the comparison of actors and lanes, none of the participants answered with the “Others” option, as such, it is possible to infer that the annotation of the

AND splits improved the interpretation of some participants concerning the tasks which can be executed in parallel in the process.

### 6.3 Discussion

In this chapter, the evaluation of the approach was performed through the means of a questionnaire, for which the main goal was to determine whether a practical application using the developed prototype would present positive results. As such, the presented questionnaire was designed aiming to evaluate four points considered as crucial for an assertive conclusion. The points covered by the analysis were in respect to the: *comprehension* of the annotated text, *correctness* of the annotation, *usefulness* for modeling purposes and *effectiveness* for interpreting the business process.

More specifically, the analysis of the three first points indicated that the annotation of business process descriptions presented in this study was well evaluated by the participants, with emphasis in its usefulness for helping in the design of the business process model. Furthermore, the evaluation of the last point has shown that, for most cases, the annotation improves the interpretation of business elements contained in the process description.

During the analysis of each of those points, valuable feedback comments were gathered and presented, which could serve as points of improvements for future works. Although this experiment has shown very promising results, it is important to note that this evaluation is restricted to one annotated process description. A large scale experiment, for instance, by using the prototype within the context of a whole organization and a diversity of business process descriptions would have a high value and provide a complete validation.

Finally, the last question of the questionnaire allowed the participants to upload their own ideal version of annotation of the presented process description. The results of the submitted documents can be seen in Appendix A and might have value for improving the prototype, specially by allowing more user customization in terms of the visual elements. Amongst the different visualization features found in the submitted documents are: underline certain annotated words and use colors in the background instead of changing the color of the words.

## 7 CASE STUDY

In this chapter, a second experiment is conducted in order to complement the evaluation performed in Chapter 6. For the context of this experiment, no external participants are involved, but rather focusing on a case study that covers one of the possible use cases of the approach, using the developed prototype for designing a business process model.

As one of the purposes of the presented approach is to help the process modeler by identifying a subset of BPMN 2.0 elements in the process description, the first part of this chapter emphasizes how the user-interactive and visual features of the prototype can lead to the creation of the process model. The second part of this chapter compares the text annotation based on a preprocessed description, according to Section 4.1, and the text annotation based on the original text description.

The demonstration of the process model design following the approach is divided in two steps: submitting the preprocessed description in the developed prototype and selecting the desired process element types to be annotated; manually creating the model based on the annotated elements. As the test set, two process descriptions originated from the *Humboldt University of Berlin* and commonly used in related researches (FRIEDRICH; MENDLING; PUHLMANN, 2011; HONKISZ; KLUZA; WIŚNIEWSKI, 2018; SILVA et al., 2019) were selected and preprocessed in accordance to the guidelines presented in Section 4.1.

In the first step, both process descriptions were annotated using the prototype developed in Chapter 5, selecting the entire set of identifiable BPMN 2.0 element types and their respective icons. Figures 7.1 and 7.2 show both process descriptions annotated.

### **Computer repair**

*When a customer brings in a defective computer, the CRS verifies the defect and hands out a repair cost calculation back. If the customer decides that the costs are acceptable, the process continues, otherwise she takes her computer home unrepaired. The ongoing repair consists of two activities, which are executed, in an arbitrary order. The first activity is to verify and adjust the hardware, while the second activity verifies and configures the software. After each of these activities, the proper system functionality is tested. If an error is detected another arbitrary repair activity is executed, otherwise the repair is finished.*



















## Hotel Service


*The Evanstonian is an upscale independent hotel. When a guest calls room service at The Evanstonian, the room-service manager takes down the order. She then submits an order ticket to the kitchen to begin preparing the food. She also gives an order to the sommelier (i.e., the wine waiter) to fetch wine from the cellar and to prepare any other alcoholic beverages. Eighty percent of room-service orders include wine or some other alcoholic beverage. Finally, she assigns the order to the waiter. While the kitchen and the sommelier are doing their tasks, the waiter readies a cart (i.e., puts a tablecloth on the cart and gathers silverware). The waiter is also responsible for nonalcoholic drinks. Once the food, wine, and cart are ready, the waiter delivers it to the guest's room. After returning to the room-service station, the waiter debits the guest's account. The waiter may wait to do the billing if he has another order to prepare or deliver.*

The second step consists of using the annotated process descriptions in order to help to design the process model. However, in the case of the two test samples used, the process descriptions don't explicitly contain all elements required to build a process model. For example, *Hotel Service* lacks an explicitly defined End Event. For such reason, the process models are built *based* on the identified elements, but not necessarily restricted to them. Moreover, whether the actors of the process should be modeled as lanes or pools is part of the process modeler's interpretation. Furthermore, in order to maintain BPMN's 2.0 guidelines, sequence flows and message flows must be included, and, for the latter, additional message events may also be needed. As a result, Figures 7.5 and 7.6 show the

Figure 7.1: Computer Repair annotated

Marked Text
 Show icons

When  a customer  brings in a defective computer,  the CRS  verifies the defect and hands out a repair cost calculation back.  [XORBRANCH] If the customer decides that the costs are acceptable, the process  continues, [XORBRANCH] otherwise she  takes her computer home unrepaired  [XORJOIN]. The ongoing repair consists of two activities, which  are executed, in an arbitrary order.  [ANDBRANCH] The first activity  is to verify and adjust the hardware, [ANDBRANCH] while the second activity  verifies and  configures the software  [ANDJOIN]. After each of these activities, the proper system functionality  is tested.  [XORBRANCH] If an error is detected another arbitrary repair activity  is executed, [XORBRANCH] otherwise the repair  is finished.




























 Change Text


Source: The Authors



Figure 7.2: Hotel Service annotated

Marked Text
 Show icons

The Evanstonian is an upscale independent hotel . When  a guest  calls room service at The Evanstonian ,  the room-service manager  takes down the order . She then  submits an order ticket to the kitchen to begin preparing the food . She also  gives an order to the sommelier ( i.e. , the wine waiter )  to fetch wine from the cellar and  to prepare any other alcoholic beverages . Eighty percent of room-service orders  include  [XORBRANCH] wine or [XORBRANCH] some other alcoholic beverage .  [XORJOIN] Finally , she  assigns the order to the waiter .  [ANDBRANCH] While  the kitchen and [ANDBRANCH]  the sommelier  are doing their tasks , [ANDBRANCH]  the waiter  readies a cart ( i.e. , puts a tablecloth on the cart and gathers silverware  [ANDJOIN] ) . The waiter is also responsible for nonalcoholic drinks . Once  [ANDBRANCH] the food , [ANDBRANCH] wine , and [ANDBRANCH] cart  are ready  [ANDJOIN] , the waiter  delivers it to the guest 's room . After  returning to the room-service station , the waiter  debits the guest 's account . The waiter  may wait to do the billing if he  has another order to prepare or deliver .

 Change Text

Source: The Authors

built process models.

In order to highlight the potential of this study, a comparison was performed between the set of process elements annotated by the approach with the set of process elements in the original process model (Figures 7.7 and 7.8). As these models were previously built by unknown modelers based on the original process descriptions, they contain elements which are not covered by the presented approach. Therefore, such elements are not considered for the purpose of comparison. As form of comparison, the standard metrics of precision and recall are used and defined as:

$$precision = \frac{\|P_{ni} \cap P_{no}\|}{\|P_{ni}\|} \quad (7.1)$$

$$recall = \frac{\|P_{ni} \cap P_{no}\|}{\|P_{no}\|} \quad (7.2)$$

where  $P_{ni}$  represents the set of process elements annotated by the presented approach and  $P_{no}$  represents the set of process elements in the original process models, disregarding BPMN 2.0 elements which are out of scope, hence, not comparable.

The calculated values of each variable for the *Computer Repair* and the *Hotel Service* examples are shown in table 7.1. The first example resulted in a precision  $\approx 0,77$  and a recall  $\approx 0,60$ . This indicates that the approach provided a precision of 77% of identified process elements that are also present in the original process model and

Table 7.1: Variable values for precision and recall

	Computer Repair	Hotel Service
$\  P_{ni} \ $	18	27
$\  P_{no} \ $	23	28
$\  P_{ni} \cap P_{no} \ $	14	14

Source: The Authors

annotated 60% of the set of comparable process elements used in the original process model. The second example resulted in a precision  $\approx 0,51$  and a recall = 0.5. This indicates that the precision of identified elements present in the original process model is 51% and that 50% of the model's set of comparable elements were annotated by the prototype.

It is important to note that these results depend on which process model is used for comparison. Due to the representative capabilities of the BPMN 2.0, there are several forms with which a process model can be designed, hence, the same business process can be modeled using different elements, execution flows, labels etc. As the original models were built by expert process modelers, it is expected that they contain considerably more elements, hence, impacting the recall, and more complex execution flows, thus, impacting the precision. In conclusion, it is presented how the approach can be used to build a semantically similar process model, however, structurally different, due to the complexity of the original process models.

Additionally, a second modeling experiment is performed, using the original (un-processed) description, meaning that the input text is not preprocessed according to the guidelines. As per Section 4.1, the input text should follow a certain structure in order to have a better accuracy of annotated elements. Nevertheless, the developed prototype does not oblige the user to utilize such structured text. As such, the purpose of this comparison is to demonstrate how effective is the presented annotation approach when using the original process description, illustrated by Figure 7.3, in comparison to the preprocessed description.

### Computer Repair (Original)

- 1 *#The workflow of a computer repair service (CRS) can be described as follows.*
- 2 *A customer brings in a defective computer and the CRS checks the defect and hands out*
- 3 *a repair cost calculation back. If the customer decides that the costs are acceptable, the*
- 4 *process continues, otherwise she takes her computer home unrepaired. The ongoing re-*

5 pair consists of two activities, which are executed, in an arbitrary order. The first activity  
 6 is to check and repair the hardware, whereas the second activity checks and configures  
 7 the software. After each of these activities, the proper system functionality is tested. If  
 8 an error is detected another arbitrary repair activity is executed, otherwise the repair is  
 9 finished.

When comparing the preprocessed description illustrated in Figure 7.1, which follows the presented guidelines, with the original *Computer Repair* process description, illustrated in Figure 7.3, some differences are notable:

- The first sentence is completely removed, as it does not contribute to the process modeling.
- Signal word “*When*” is added in line 2 to indicate the absence of time duration of the action.
- The verb “*check*” in lines 2 and 6 is replaced by the synonym “*verify*”, in order to avoid ambiguous part of speech definitions.
- The signal word “*whereas*” in line 6 is replaced by the signal word “*while*”, as the first indicates exclusivity and the second indicates concurrency.

The result of these changes are demonstrated by comparing the identified elements in Figure 7.3 with respect to the identified elements in Figure 7.1 and directly reflects on the importance of following the guidelines to create a preprocessed description. This

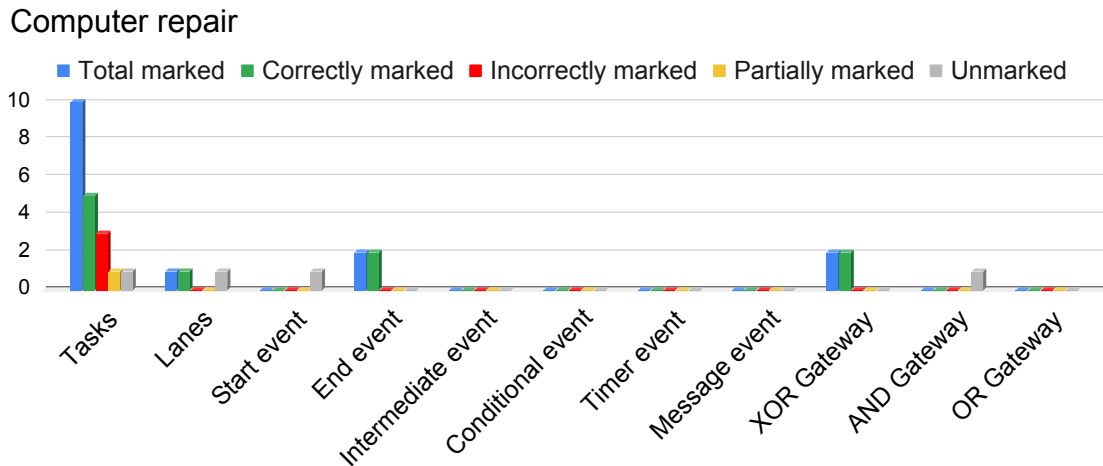
Figure 7.3: Computer Repair (original) annotated

Marked Text
 Show icons

#The workflow of a computer repair service ( CRS ) can  **be described** as  **follows** .  
 **A customer**  **brings in a defective computer and the CRS**  **checks the defect and hands** out a repair cost calculation back .  [XORBRANCH] **If the customer decides** that the costs are acceptable , the process  **continues** , [XORBRANCH] **otherwise** she  **takes her computer home unrepaired**  [XORJOIN] . The ongoing repair consists of two activities , which  **are executed** , in an arbitrary order . The first activity  **is to check and repair the hardware** , whereas the second activity  **checks** and configures the software . After each of these activities , the proper system functionality  **is tested** .  [XORBRANCH] **If an error is detected** another arbitrary repair activity  **is executed** , [XORBRANCH] **otherwise** the repair  **is finished** .

Source: The Authors

Figure 7.4: Results of identified elements in the original text



Source: The Authors

comparison is performed on each element type of the identifiable BPMN 2.0 elements and is split into five categories.

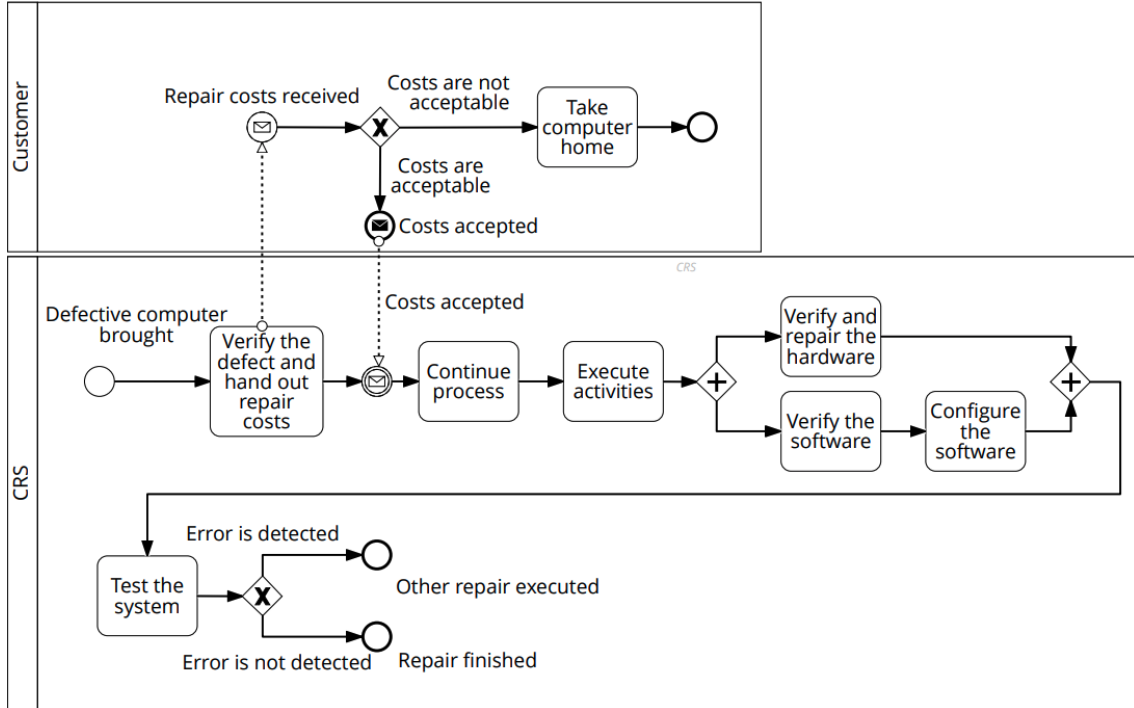
The *Total marked* category represents the amount of the respective process element which was annotated in the original process description. *Correctly marked* stands for the amount of the respective process element which was annotated both in the preprocessed description and in the original description. Oppositely, *Incorrectly marked* stands for the amount of the respective process element which was annotated exclusively in the original process description. The *Partially Marked* category represents the amount of the respective process element which was annotated in the original description, but missed a complementary information in the text, such as an object. Finally, *Unmarked* stands for the amount of the respective process element which was annotated exclusively in the preprocessed description.

As illustrated in Figure 7.4, the Tasks show the highest discrepancy. The incorrectly marked tasks refer to the first line of the original process description, which was removed in the preprocessed description. The partially marked and unmarked Tasks are both consequences of the ambiguous verb “check”, which misleads the text processing algorithms. The lack of the “When” conjunction causes both the unmarked Lanes and Start Event. The unmarked AND Gateway is due to the “whereas” signal word in the original description, which does not indicate a parallel execution.

Finally, the precision and recall are calculated based on equations 7.1 and 7.2. For this context,  $P_{ni}$  is defined as the set of annotated process elements in the original process description and  $P_{no}$  is defined as the set of annotated process elements in the preprocessed description. The correctly marked process elements are considered for the intersection of

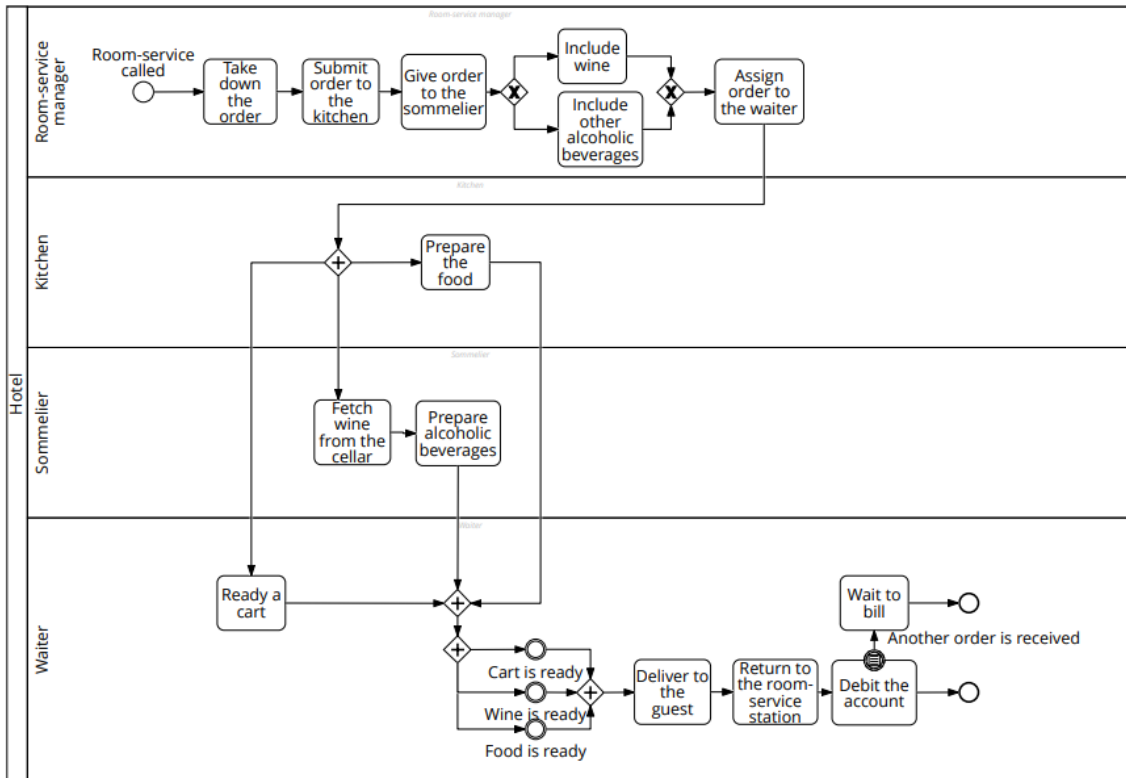
both sets. The annotation of process elements in the original process description has a precision of 55,5% and a recall of 62,5% in comparison with the annotated process elements in the preprocessed description. While the precision is low, due to the removed unnecessary information at the beginning of the original process description, the recall indicates that most elements can still be correctly identified.

Figure 7.5: Computer Repair process model



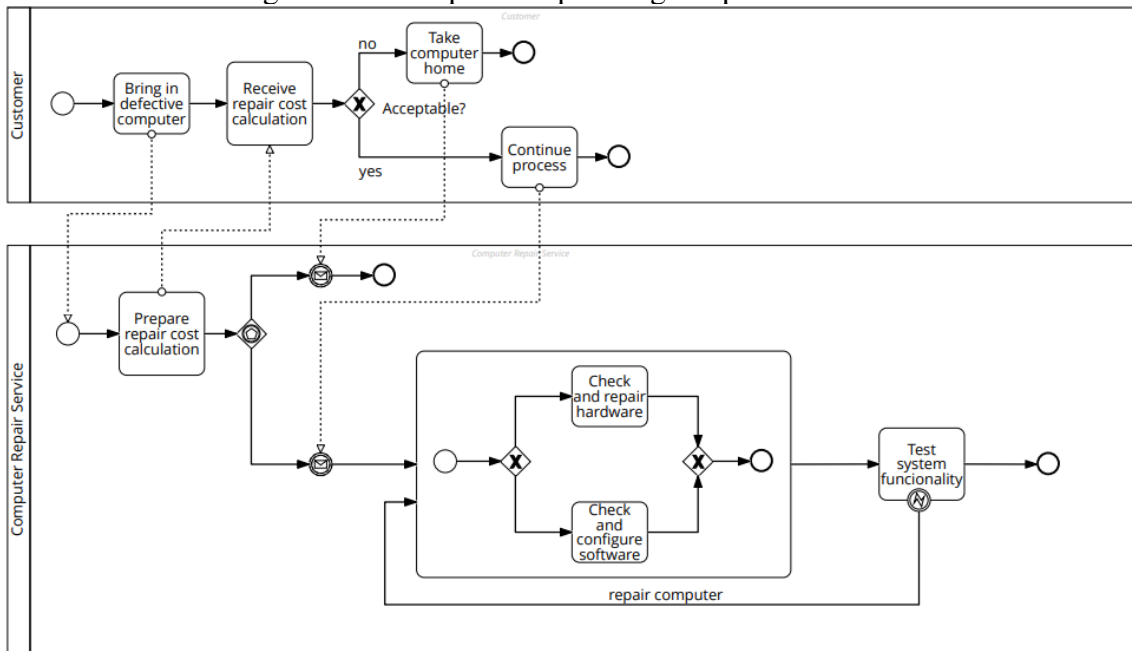
Source: The Authors

Figure 7.6: Hotel Service process model



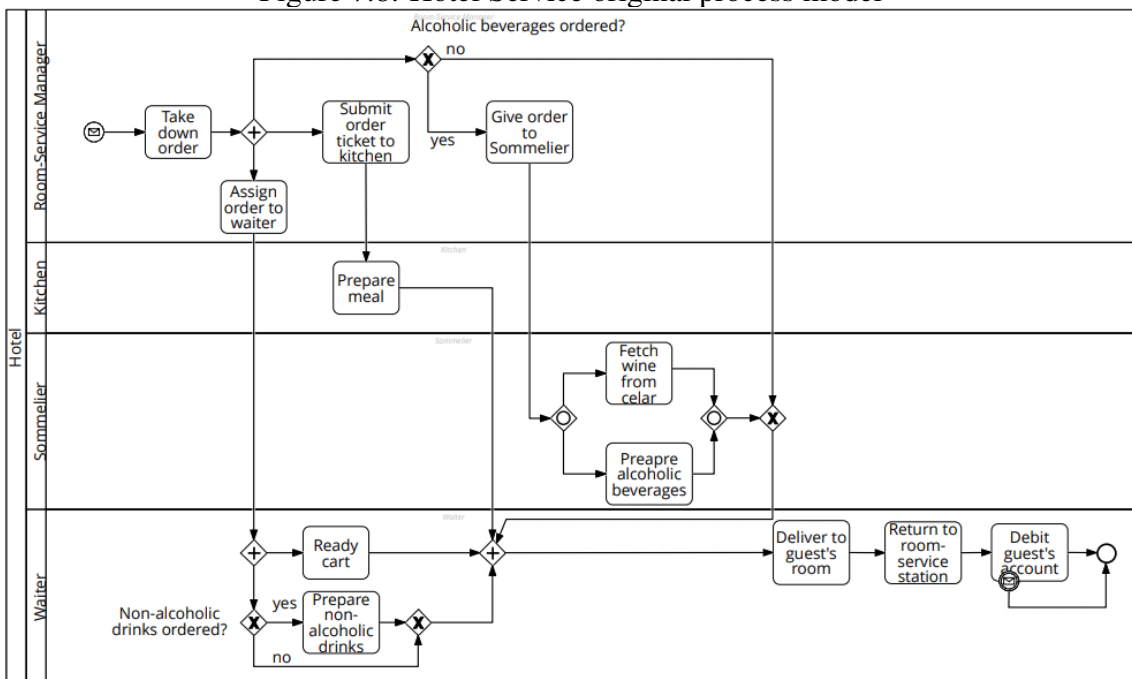
Source: The Authors

Figure 7.7: Computer Repair original process model



Source: Humboldt University of Berlin

Figure 7.8: Hotel Service original process model



Source: Humboldt University of Berlin

## 8 CONCLUSION

In this study, an approach to identify, extract and visually annotate process elements in process descriptions was presented. The hypothesis proposed in Chapter 1, questioned whether a mixed notation based on the process description annotated by visual features, including symbols of the BPMN 2.0, would assist in the comprehension of the process. As demonstrated in Section 6.2, the annotated process description not only assisted the interpretation of the participants in the questionnaire, but also improved it.

The presented approach combines NLP techniques, text analyzing algorithms and a structured data format enhancing the flexibility for visual representation of extracted business process elements. Although being technology independent, the secondary motivation of building a functional implementation with extensible technologies was achieved and detailed in Chapter 5. The shown prototype transposed and adapted the prototype created in the state of the art study of Friedrich et. al. using integrated modern technologies in order to fit in with the further phases of the approach.

In particular, Section 5.1 detailed how the prototype introduced a separation of concerns between the processing and the displaying phases of the presented approach, by using a client-server model. Additionally, Section 5.3 showed how the Python based server component communicated with the Stanford CoreNLP, the successor implementation of the Stanford Parser, in order to leverage its NLP pipeline. Furthermore, the detailed implementation of the client component in Section 5.4 showed how *ReactJS* can leverage from several external modular components in order to achieve the user interactive visual display of the annotated text.

As briefly mentioned throughout the study and further encouraged by the results in Chapter 6, the presented approach can be useful for several use cases, such as: assisting the design of process modeling, identification of the absence of certain process elements in the process description, visualization of a restricted sub-set of elements (e.g., only *Tasks* and *Events*), auditing business processes, teaching and training purposes, and others. Furthermore, to the best of the authors' knowledge, this study is a pioneer on visually highlighting the original process descriptions of process models and, additionally, allowing user-interactive annotations.

As it can be observed during the evaluation of the approach, the annotated text can provide many insights on which elements should be present in a process model but does not exclude the interpretation of the process modeler. Therefore, as a future exper-



iment, the approach will be evaluated with a variety of process modelers. Furthermore, the presented guidelines to describe a business process improve the efficiency of the NLP techniques, however, this is a procedure that is performed manually and can be time consuming depending on the size and complexity of the process description. In that regard, extending this approach to automate the creation of a process description which follows the guidelines can be a valuable contribution.

In addition, as mentioned in Section 6.2, the approach used for extracting the process elements could be enhanced with more accurate text parsing models, replacing the statistical based model for the current state of the art models based on neural networks and shift-reduce methods. Such enhancement could help to avoid the inconsistencies of the annotated tasks and may even mitigate some ambiguity issues, leading to less manual preprocessing in the process description. During the testing phase of the presented prototype, 4 different models were tested, however, it was noted that the structure of the constituency and dependency parsing was significantly different, impacting the sentence analysis algorithms. Therefore, such modification would require an extensive study to adapt the whole elements extraction phase.

The feedback comments obtained in the evaluation of the approach also presented valuable insights of improvements in the visual annotation. In particular, different visualization features were suggested for the branches of gateways, for example, using indentation or surrounding the respective sentences with special characters. Furthermore, insights were also gathered concerning the interactivity of the prototype, for example, allowing the end user to choose whether the colors are added in the text or in the background. In the same topic, a helpful improvement, although extensive, would be to allow the user to directly edit the annotation metadata, providing the capability of changing the start and end indices of the annotated element.

Besides the mentioned points of improvements, this study can be used as ground for future researches, not restricted to the field of process discovery. As an example, the monitoring of the execution of certain processes could be performed with the assistance of the generated metadata containing the process elements. Furthermore, the annotation could be particularly useful when combined with corporate documentation mining tools, using information contained in wiki pages, backlogs, and others, in order to find gaps regarding process alignment.

In conclusion, as it stands, the presented approach has shown promising results and multiple use cases. Moreover, the encapsulation of the results of the extraction and

identification of process elements in the form of a widely used notation (i.e, JSON) provides the flexibility for further consumption of other automated tools which may have a variety of different purposes. As such, the value of this approach is not restricted to the scope of this study, but rather encourages the emergence of complementary researches.

## REFERENCES

- AALST, W. V. D. Process mining. **Communications of the ACM**, ACM New York, NY, USA, v. 55, n. 8, p. 76–83, 2012.
- AGGARWAL, S. Modern web-development using reactjs. **International Journal of Recent Research Aspects**, v. 5, n. 1, p. 2349–7688, 2018.
- ALLEN, I. E.; SEAMAN, C. A. Likert scales and data analyses. **Quality progress**, v. 40, n. 7, p. 64–65, 2007.
- ALLWEYER, T. **Introduction to the Standard for Business Process Modeling**. [S.l.]: Germany: Herstellung und Verlag: Books on Demand GmbH, 2009.
- ALONSO, G. et al. Web services. In: **Web services**. [S.l.]: Springer, 2004. p. 123–149.
- AYSOLMAZ, B. et al. A semi-automated approach for generating natural language requirements documents based on business process models. **Information and Software Technology**, Elsevier, v. 93, p. 14–29, 2018.
- BIRD, E. L. **Steven and Ewan Klein. 2009. Natural Language Processing with Python**. [S.l.]: O’Reilly Media Inc, 2009.
- BLUMBERG, R.; ATRE, S. The problem with unstructured data. **Dm Review**, POWELL PUBLISHING INC, v. 13, n. 42-49, p. 62, 2003.
- CHOMSKY, N.; LIGHTFOOT, D. W. **Syntactic structures**. [S.l.]: Walter de Gruyter, 2002.
- COMMISSION, I. O. for S. E. et al. Iso/iec 19510: 2013. **Information Technology–Object Management Group Business Process Model and Notation**, 2013.
- DUMAS, M. et al. **Fundamentals of Business Process Management**. [S.l.]: Springer, 2018.
- ERL, T. **SOA Design Patterns (paperback)**. [S.l.]: Pearson Education, 2008.
- FEDOSEJEV, A. **React. js essentials**. [S.l.]: Packt Publishing Ltd, 2015.
- FERREIRA, R. C. B. et al. Recognition of business process elements in natural language texts. In: SPRINGER. **International Conference on Enterprise Information Systems**. [S.l.], 2017. p. 591–610.
- FIELDING, R. T.; TAYLOR, R. N. **Architectural styles and the design of network-based software architectures**. [S.l.]: University of California, Irvine Irvine, 2000.
- FRIEDRICH, F.; MENDLING, J.; PUHLMANN, F. Process model generation from natural language text. In: SPRINGER. **International Conference on Advanced Information Systems Engineering**. [S.l.], 2011. p. 482–496.
- HEINEMAN, G. T.; COUNCILL, W. T. Component-based software engineering. **Putting the pieces together, addison-westley**, Springer, p. 5, 2001.

HONKISZ, K.; KLUZA, K.; WIŚNIEWSKI, P. A concept for generating business process models from natural language description. In: SPRINGER. **International Conference on Knowledge Science, Engineering and Management**. [S.l.], 2018. p. 91–103.

HOVLAND, P. et al. A quality-of-service architecture for high-performance numerical components. In: **Proceedings of the Workshop on QoS in Component-Based Software Engineering**. [S.l.: s.n.], 2003.

ICONIFY. **Iconify React Repository**. 2019. Available from Internet: <<https://github.com/iconify/iconify-react>>. Accessed in: 22 sep. 2020.

INDURKHYA, N.; DAMERAU, F. J. **Handbook of natural language processing**. [S.l.]: CRC Press, 2010.

IVANCHIKJ, A.; SERBOUT, S.; PAUTASSO, C. From text to visual bpmn process models: design and evaluation. In: **Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems**. [S.l.: s.n.], 2020. p. 229–239.

KETTINGER, W. J.; TENG, J. T.; GUHA, S. Business process change: a study of methodologies, techniques, and tools. **MIS quarterly**, JSTOR, p. 55–80, 1997.

KLEIN, D.; MANNING, C. D. Fast exact inference with a factored model for natural language parsing. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2003. p. 3–10.

LAPLANTE, P. A. **What every engineer should know about software engineering**. [S.l.]: CRC Press, 2007.

LEOPOLD, H. et al. Searching textual and model-based process descriptions based on a unified data format. **Software & Systems Modeling**, Springer, v. 18, n. 2, p. 1179–1194, 2019.

LEOPOLD, H.; MENDLING, J.; POLYVYANYYY, A. Generating natural language texts from business process models. In: SPRINGER. **International Conference on Advanced Information Systems Engineering**. [S.l.], 2012. p. 64–79.

LIDDY, E. D. **Natural language processing**. 2001.

LIKERT, R. A technique for the measurement of attitudes. **Archives of psychology**, 1932.

LYONS, J. **Natural Language and Universal Grammar: Volume 1: Essays in Linguistic Theory**. [S.l.]: Cambridge University Press, 1991.

MALIK, S.; BAJWA, I. S. Back to origin: Transformation of business process models to business rules. In: SPRINGER. **International Conference on Business Process Management**. [S.l.], 2012. p. 611–622.

MANNING, C.; SCHUTZE, H. **Foundations of statistical natural language processing**. [S.l.]: MIT press, 1999.

- MANNING, C. D. et al. The stanford corenlp natural language processing toolkit. In: **Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations**. [S.l.: s.n.], 2014. p. 55–60.
- MARNEFFE, M.-C. D. et al. Generating typed dependency parses from phrase structure parses. In: **Lrec**. [S.l.: s.n.], 2006. v. 6, p. 449–454.
- MEITZ, M.; LEOPOLD, H.; MENDLING, J. An approach to support process model validation based on text generation. In: **EMISA Forum**. [S.l.: s.n.], 2013. v. 33, n. 2, p. 7–20.
- MENDLING, J.; REIJERS, H. A.; AALST, W. M. van der. Seven process modeling guidelines (7pmg). **Information and Software Technology**, Elsevier, v. 52, n. 2, p. 127–136, 2010.
- MILLMAN, K. J.; AIVAZIS, M. Python for scientists and engineers. **Computing in Science & Engineering**, IEEE, v. 13, n. 2, p. 9–12, 2011.
- MUEHLEN, M. Z.; RECKER, J. How much language is enough? theoretical and practical use of the business process modeling notation. In: **Seminal Contributions to Information Systems Engineering**. [S.l.]: Springer, 2013. p. 429–443.
- OLIPHANT, T. E. Python for scientific computing. **Computing in Science & Engineering**, IEEE, v. 9, n. 3, p. 10–20, 2007.
- OTTENSOOSER, A. et al. Making sense of business process descriptions: An experimental comparison of graphical and textual notations. **Journal of Systems and Software**, Elsevier, v. 85, n. 3, p. 596–606, 2012.
- QIAN, C. et al. An approach for process model extraction by multi-grained text classification. In: SPRINGER. **International Conference on Advanced Information Systems Engineering**. [S.l.], 2020. p. 268–282.
- REITER, E.; DALE, R. **Building natural language generation systems**. [S.l.]: Cambridge university press, 2000.
- RIEFER, M.; TERNIS, S. F.; THALER, T. Mining process models from natural language text: A state-of-the-art analysis. **Multikonferenz Wirtschaftsinformatik (MKWI-16), March**, p. 9–11, 2016.
- RODRIGUES, R. D. A.; AZEVEDO, L. G.; REVOREDO, K. C. Bpm2text: A language independent framework for business process models to natural language text. **iSys-Revista Brasileira de Sistemas de Informação**, v. 9, n. 4, p. 38–56, 2016.
- SEARS, A.; JACKO, J. A. **Human-computer interaction fundamentals**. [S.l.]: CRC Press, 2009.
- SHENG, Q. Z. et al. Web services composition: A decade's overview. **Information Sciences**, Elsevier, v. 280, p. 218–238, 2014.
- SHNEIDERMAN, B. et al. **Designing the user interface: strategies for effective human-computer interaction**. [S.l.]: Pearson, 2016.

SHUTE, Z. **Advanced JavaScript: speed up web development with the powerful features and benefits of JavaScript**. [S.l.]: Packt Publishing Ltd, 2019.

SILVA, T. S. et al. A service-oriented architecture for generating sound process descriptions. In: IEEE. **2019 IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC)**. [S.l.], 2019. p. 1–10.

SILVA, T. S. et al. Empirical analysis of sentence templates and ambiguity issues for business process descriptions. In: SPRINGER. **OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"**. [S.l.], 2018. p. 279–297.

SWELLER, J.; MERRIENBOER, J. J. V.; PAAS, F. G. Cognitive architecture and instructional design. **Educational psychology review**, Springer, v. 10, n. 3, p. 251–296, 1998.

THOM, L. H.; REICHERT, M.; IOCHPE, C. Activity patterns in process-aware information systems: Basic concepts and empirical evidence. **International Journal of Business Process Integration and Management**, Inderscience Publishers, v. 4, n. 2, p. 93–110, 2009.

VIPUL, A.; SONPATKI, P. **ReactJS by Example-Building Modern Web Applications with React**. [S.l.]: Packt Publishing Ltd, 2016.

ZENG, Q. et al. Missing procedural texts repairing based on process model and activity description templates. **IEEE Access**, IEEE, v. 8, p. 12999–13010, 2020.

ZIMOCH, M. et al. The repercussions of business process modeling notations on mental load and mental effort. In: SPRINGER. **International Conference on Business Process Management**. [S.l.], 2018. p. 133–145.

## APPENDIX A — ANNOTATIONS SUBMITTED IN THE QUESTIONNAIRE

Figure A.1: Participant annotation 1

When a customer brings in a defective computer, the CRS verifies the defect and hands out a repair cost calculation back.

If the customer [XORSPLIT] decides that the costs are acceptable, the process continues, [XORSPLIT] otherwise, she takes her computer home unrepaired.

The ongoing repair consists of two activities, which are executed, in an arbitrary order.

The first activity is [ANDSPLIT] to verify and adjust the hardware, while the second activity [ANDSPLIT] verifies and configures the software [ANDJOIN] finishing the “individual repair” (originally missing information).

After each of these activities, the proper system functionality is tested. [XORSPLIT] If an error is detected another arbitrary repair activity is executed, [XORSPLIT] otherwise, the repair is finished.



















Source: Anonymous Participant

Figure A.2: Participant annotation 2

When a customer brings in a defective computer, the CRS verifies the defect and hands out a repair cost calculation back. [XORBRANCH] If the customer decides that the costs are acceptable, the process continues, [XORBRANCH] otherwise, she takes her computer home unrepaired [XORJOIN]. The ongoing repair consists of two activities, which are executed, in an arbitrary order. [ANDBRANCH]The first activity is to verify and adjust the hardware, [ANDBRANCH] while the second activity verifies and configures the software[ANDJOIN]. After each of these activities, the proper system functionality is tested. [XORBRANCH] If an error is detected another arbitrary repair activity is executed, [XORBRANCH] otherwise, [XORJOIN] the repair is finished

Source: Anonymous Participant

Figure A.3: Participant annotation 3

○ When a  customer  brings in a defective computer, the  CRS  verifies the defect and  hands out a repair cost calculation back.  [XORBRANCH] If the customer decides that the costs are acceptable, the  process continues, [XORBRANCH] otherwise, she  takes her computer home unrepaired  [XORJOIN]. The ongoing repair consists of two activities, which are executed, in an arbitrary order.  [ANDBRANCH] The first activity is to  verify and  adjust the hardware, [ANDBRANCH] while the second activity  verifies and  configures the software.  [ANDJOIN]. After each of these activities, the proper system functionality is  tested.  [XORBRANCH] If an error is detected another arbitrary  repair activity ○ is executed, [XORBRANCH] otherwise, the repair ○ is finish.

Source: Anonymous Participant

Figure A.4: Participant annotation 4

When a customer brings in a defective computer, the CRS verifies the defect and hands out a repair cost calculation back. [XORBRANCH] If the customer decides that the costs are acceptable, the process continues, otherwise, she takes her computer home unrepaired [XORJOIN]. [ANDBRANCH] The ongoing repair consists of two activities, which are executed, in an arbitrary order. The first activity is to verify and adjust the hardware, while the second activity verifies and configures the software. [ANDJOIN] After each of these activities, the proper system functionality is tested. [XORBRANCH] If an error is detected another arbitrary repair activity is executed, otherwise, the repair is finished [ANDJOIN]

Source: Anonymous Participant



Figure A.5: Participant annotation 5

When a customer ○brings in a defective computer, the CRS □verifies the defect and □hands out a repair cost calculation back. [XORBRANCH] If the customer decides that the costs are acceptable, the process continues,[XORBRANCH] otherwise, she □takes her computer home unrepaired. The ongoing repair consists of two activities, which are executed, in an arbitrary order. [ANDBRACH]The first activity is to □verify and □adjust the hardware, while [ANDBRACH] the second activity □verifies and □configures the software [ANDJOIN]. After each of these activities, the □proper system functionality is tested. [XORBRANCH] If an error is detected □another arbitrary repair activity is executed, [XORBRANCH] otherwise, the repair is finished [XORJOIN] [XORJOIN] ●.

Source: Anonymous Participant

Figure A.6: Participant annotation 6

When a customer brings in a defective computer, the CRS verifies the defect and hands out a repair cost calculation back. If the customer decides that the costs are acceptable, the process continues, otherwise, she takes her computer home unrepaired. The ongoing repair consists of two activities, which are executed, in an arbitrary order. The first activity is to verify and adjust the hardware, while the second activity verifies and configures the software. After each of these activities, the proper system functionality is tested. If an error is detected another arbitrary repair activity is executed, otherwise, the repair is finished

Source: Anonymous Participant

Figure A.7: Participant annotation 7

When a customer brings in a defective computer, the CRS verifies the defect and hands out a repair cost calculation back. If the customer decides that the costs are acceptable, the process continues, otherwise, she takes her computer home unrepaired. The ongoing repair consists of two activities, which are executed, in an arbitrary order. The first activity is to verify and adjust the hardware, while the second activity verifies and configures the software. After each of these activities, the proper system functionality is tested. If an error is detected another arbitrary repair activity is executed, otherwise, the repair is finished

Source: Anonymous Participant

Figure A.8: Participant annotation 8

When a customer ○brings in a defective computer, the CRS verifies the defect and hands out a repair cost calculation back.[XORBRANCH] If the customer decides that the costs are acceptable, the process continues, [XORBRANCH] otherwise, she takes her computer home unrepaired. The ongoing repair consists of two activities, which are executed, in an arbitrary order. [ANDBRANCH] The first activity is to verify and adjust the hardware, [ANDBRACH] while the second activity verifies and configures the software [ANDJOIN]. After each of these activities, the proper system functionality is tested. [XORBRANCH] If an error is detected another arbitrary repair activity is executed, [XORBRANCH] otherwise, the repair is finished [XORJOIN] ○.

Source: Anonymous Participant