UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

THANNER SOARES SILVA

An architecture for enabling business process-oriented text generation

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. Lucinéia Heloisa Thom

Porto Alegre
April 2019

CIP — CATALOGING-IN-PUBLICATION

Silva, Thanner Soares

An architecture for enabling business process-oriented text generation / Thanner Soares Silva. – Porto Alegre: PPGC da UFRGS, 2019.

153 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2019. Advisor: Lucinéia Heloisa Thom.

1. Business process management. 2. Business process model and notation. 3. Natural language processing. 4. Service oriented architecture. I. Heloisa Thom, Lucinéia. II. Título.

ABSTRACT

Business process management has become an increasingly present activity in organizations. In this context, business process descriptions are considered as a useful artifact in both identifying business processes and complementing business process documentation. However, organizations do not always create business process descriptions to document their business processes. In addition, business process descriptions may not follow a specific format, which may lead to contain ambiguous or non-recurring sentences that make it difficult to understand the process. Thus, this dissertation aims to develop an approach that enables the generation of business process-oriented texts. In the context of this work, the business process-oriented text is defined as a text that is structured, able to maintain the maximum information related to the business process, and able to check the quality of the process in relation to the BPMN 2.0 and in relation to soundness. In order to achieve this goal, was performed an analysis in the literature and in 64 business process descriptions in order to define how business process-oriented texts should be. In addition, an SOA-based architecture was developed in which the steps required to generate the business process-oriented text are addressed to the individual services. The analysis made it possible to find 101 recurring sentence templates in business processes descriptions, of which 13 were considered to have ambiguity issues based on the adopted criteria. Furthermore, a prototype was developed and the process description produced by the approach was compared to the original process model through a process similarity technique. The findings made in order to define how the business process-oriented texts should be can support other approaches in generating business process descriptions more suitable for process analysts and domain experts. Finally, the architecture can be enhanced with other services capable of providing other functionalities that contribute to the creation and management of business processes descriptions in organizations.

Keywords: Business process management. business process model and notation. natural language processing. service oriented architecture.

Uma arquitetura para geração de texto orientado a processos de negócio

RESUMO

O gerenciamento de processos de negócio tem se tornado uma atividade cada vez mais presente nas organizações. Nesse contexto, descrições de processos de negócio são consideradas um artefato útil na identificação de processos de negócio e na complementação da documentação de processos de negócio. No entanto, as organizações nem sempre criam descrições de processos de negócio para documentar seus processos de negócio. Além disso, descrições de processos de negócio podem não seguir um formato específico, podendo conter sentenças ambíguas ou não recorrentes, que dificultam a compreensão do processo. Assim, esta dissertação visa desenvolver uma abordagem que permita a geração de textos orientados a processos de negócio. No contexto deste trabalho, o texto orientado a processos de negócio é definido como um texto estruturado, capaz de manter o máximo de informações relacionadas ao processo de negócio, e capaz de verificar a qualidade do processo em relação ao BPMN 2.0. e em relação ao *soundness*. Para atingir esse objetivo, foi feita uma análise na literatura e em 64 descrições de processos de negócio, a fim de definir como deveriam ser os textos orientados a processos de negócio. Além disso, foi desenvolvida uma arquitetura baseada em SOA na qual as etapas necessárias para gerar o texto orientado a processos de negócio são endereçadas a serviços individuais. A análise permitiu encontrar 101 *templates* de sentença recorrentes em descrições de processos de negócio, sendo 13 deles considerados como tendo problemas de ambiguidade com base nos critérios adotados. Além disso, um protótipo foi desenvolvido e a descrição de processo produzida pela abordagem foi comparada com o modelo de processo original através de uma técnica de similaridade de processos. As descobertas feitas para definir como devem ser os textos orientados a processos de negócio podem suportar outras abordagens na geração de descrições de processos de negócio mais adequadas para analistas de processo e especialistas de domínio. Por fim, a arquitetura pode ser aprimorada com outros serviços capazes de fornecer outras funcionalidades que contribuem para a criação e gerenciamento de descrições de processos de negócio para as organizações.

Palavras-chave: Gerenciamento de processos de negócio, modelagem e notação de processos de negócio, processamento de linguagem natural, arquitetura orientada a serviços.

LIST OF ABBREVIATIONS AND ACRONYMS

BPM	Business Process Management
BPMN	Business Process Model and Notation
DSYNT	Deep Syntactic Tree
JSON	JavaScript Object Notation
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding
OMG	Object Management Group
REST	Representational State Transfer
RPST	Refined Process Structure Tree
SBVR	Semantics of Business Vocabulary and Rules
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
WADL	Web Application Description Language
WSDL	Web Service Description Language
XML	Extensible Markup Language
XSD	XML Schema Definition
YAWL	Yet Another Workflow Language

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

In order to stay competitive, organizations document and manage their business processes. A business process can be defined as “a collection of events, activities and decisions that collectively lead to an outcome that brings value to an organization’s customers” (DUMAS et al., 2018). These processes, in turn, can be presented through different representations, such as: business process descriptions or business process models. The combination of distinct representations can improve understanding of the process as they provide different perspectives on it (NAWROCKI et al., 2006; OTTENSOOSER et al., 2012).

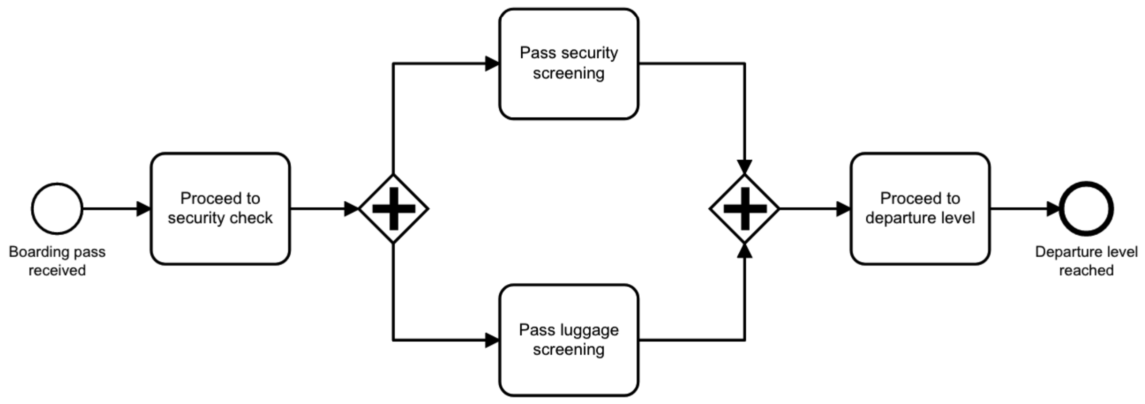
While business process descriptions are presented as natural language texts that describe how the process should occur, business process models use notations made up of visual elements and rules to represent the business process. There are several process modeling notations in the literature that can be used to model business processes. Zimoch et al. (2018) presents some: Business Process Modeling Notation (BPMN 2.0) (OMG, 2013), Declarative Process Modeling (AALST; PESIC; SCHONENBERG, 2009), eGantt Chart (LANZ; KOLB; REICHERT, 2013), Event-driven Process Chain (EPC) (AALST, 1999), Flow Chart (SCHULTHEISS; HEILIGER, 1963), Integrated Definition for Process Description Capture Method (IDEF3) (MAYER et al., 1995), Petri Net (MURATA, 1989), and UML Activity Diagram (DUMAS; HOFSTEDE, 2001).

Figure 1.1 presents a business process model created following the BPMN 2.0. This figure demonstrates a process of security check at the airport. This process starts when the *boarding pass is received*. Then, the passenger must *proceed to the security check*. There, the passenger must *pass through the security screening* and *the luggage must pass through the luggage screening*. Having the former two activities executed, the passenger must *proceed to the departure level*. This process ends when the *passenger reaches at the departure level*.

The business process model presented in Figure 1.1 is composed of different elements defined as *process elements*. Each process element has its own graphic representation and a certain semantics. As an example, the rectangle with rounded corners and with the label “Proceed to security check” corresponds to an *activity* and aims to express that the action “Proceed to security check” should be performed by some resource (in this case, the passenger).

To achieve the goal of documenting and managing their processes, organizations

Figure 1.1: Security check airport process.



Source: The authors, adapted from Dumas et al. (2018)

need to discover their processes. Dumas et al. (2018) describes the discovery of processes as the act of gathering information about existing processes and organizing them in terms of a *as-is* process models. The *as-is* process model depicts how the process currently occurs in the organization.

According to Dumas et al. (2018), the discovery of processes usually involves the participation of two main roles: the *process analyst* and the *domain specialist*. The process analyst is defined as the person who has in-depth knowledge of business process modeling techniques and uses this to construct process models. However, the process analyst usually has no knowledge of the operation of the process that will model. On the other hand, the domain specialist has detailed knowledge of the operation of the process, but has no experience with process modeling techniques.

The process discovery can be done through the use of different methods. Dumas et al. (2018) presents three different methods for process discovery, being: *interview-based discovery*, *workshop-based discovery*, and *evidence-based discovery*.

Interview-based discovery refers to methods that make up the interview with domain experts about how a process is run. Such methods involve asking questions from domain experts or even asking them to write a text describing the process.

Evidence-based discovery, in turn, covers the techniques of document analysis, observation, and automatic process discovery. The document analysis makes use of the different documents of the organization to seek to understand the process. In this context, different manuals, reports, business policies, books, forms, letters and e-mails can be used as a source by a process analysts. On the other hand, observation can be considered as a daily analysis of how people in the organization carry out their activities. Automatic process discovery, in turn, make use of event logs provided by information systems in order to find the business processes (AALST; WEIJTERS; MARUSTER, 2004).

Finally, the *workshop-based discovery* seeks to bring together different domain experts to get a more holistic view of the process. For the better understanding of the process, several sessions involving different participants may be necessary, requiring a good preparation and scheduling. Moreover, in addition to process analysts and domain experts, this method relies on other roles such as the facilitator (responsible for organizing participants' verbal contributions), the tool operator (responsible for inserting the results of the discussion in the modeling tool), and the process owner (responsible for the efficient and effective operation of the process).

This work focuses *on the creation of process descriptions so that it can contribute to process discovery (i.e. document analysis) and process documentation.*

1.1 Motivation

This work arose from a need observed in previous works of the same research group (FERREIRA; THOM; FANTINATO, 2017; FERREIRA et al., 2017). Although these works have different objectives and methodologies in relation to this work, both aim to contribute to the process discovery from business process descriptions.

In terms of relevance, this work is justified by three different reasons. First, this work allows the structuring of documents that contain information related to business processes and thus facilitate the process discovery in textual descriptions. Facilitating the process discovery in textual descriptions is desirable since the acquisition of a process model (i.e., the realization of process discovery) may require 60% of the total time spent in a workflow project (HERBST; KARAGIANNIS, 1999). Second, this work can aid in communication between process analysts and domain experts. Having a structured document describing the process can help in the interaction between these roles during process discovery. Finally, this work can serve for the understanding of the process by the actors of the organization acting, in this way, as a script to be followed by professionals.

Concerning the structuring of documents, the technique of analyzing organizational documents can be complicated since many of the organizational information is stored in unstructured form (BLUMBERG; ATRE, 2003; MAQBOOL et al., 2018). In addition, most of the available documentation of an organization's operations is not organized in a process-oriented manner (DUMAS et al., 2018).

Regarding communication between process analysts and domain experts, only process analysts understand the process models in detail (OTTENSOOSER et al., 2012; LEOPOLD; MENDLING; POLYVYANYY, 2012; MAQBOOL et al., 2018), which may hinder the validation of the process model. In this context, the use of textual descriptions could improve the understanding of the process by the domain experts (OTTENSOOSER et al., 2012) and, consequently, the interaction between these and process analysts during process model validation.

In relation to act as a script to be followed by professionals, business process descriptions can also be maintained along with process models to make process information accessible to various stakeholders, including those who are unfamiliar with the reading and interpretation of process models (AA; LEOPOLD; REIJERS, 2017). In an experiment conducted with 196 students that sought to compare the ability to understand processes from textual descriptions and process models demonstrated that the most effective approach to comprehension occurred when textual descriptions were used followed by presentation of their respective process models (OTTENSOOSER et al., 2012).

Moreover, many organizations supplement their documents with process descriptions to add information that is not demonstrated in their process models (LEOPOLD et al., 2016). This need is also observed in other areas that involve the use of models for data representation. Lavoie and Rambow (1997), in his work on generating text from object-oriented data model, comments that graphics needs to be complemented by an alternate view of the data in order to improve communication. More specifically, the authors suggest that the data could be complemented with standard english text.

Although there is an interest in creating business process descriptions, its creation involves different challenges. Among them, it is possible to highlight: *the ambiguity present in the natural language texts*, *the identification of process elements* and *the verification of the process being described*.

Referring to the ambiguity present in natural language texts, the texts can contain snippets that generate multiple interpretations. As an example, in the sentence "Then, the passenger must pass through the security screening and the luggage must pass through the luggage screening", the snippet "and" may raise doubts whether actions "pass through the security screening" and "pass through the luggage screening" should be executed in sequence (i.e., the second action is performed only after the first action has been performed) or in parallel (i.e., the two actions are performed at the same time). Such interpretations can lead to misunderstandings of the process and, consequently, to possible errors (FERRARI et al., 2017).

Concerning identification of process elements in process descriptions, in many situations people may have difficulty in identifying all process elements (e.g. events, activities) being presented in a process description. This issue has already been highlighted in other works and is presented as a point to be solved (LEOPOLD; MENDLING; POLYVYANYY, 2014). A possible solution could be to mark the elements of the process in the process descriptions and thus to help the readers to identify the elements (FERREIRA et al., 2017).

In relation to the verification of the process being described, it is interesting that not only is the description correct, but also that the process being described presents no problems (e.g., process that describes a loop that never ends, activity that is never performed, activity that does not present who is the resource that performs it). In this context, the description can be checked to conform to some modeling notation. In addition, a verification can ensure that the process has soundness (AALST, 1998). A process is considered sound if it has at least one option to finish, that at the end of the process there are no tasks still in progress, and that has no paths that will never be executed.

1.2 Goals and Hypothesis

Considering the reasons and the challenges previously presented, structured documents capable of describing processes objectively are shown as desirable artifacts for organizations. Thus, this dissertation aims to *develop an approach that allows the generation of business process-oriented text from natural language text*. In the context of this work, a business process-oriented text is defined as a text that is:

1. Structured.
2. Able to maintain the maximum information related to the business process.
3. Able to check the quality of the process in relation to the BPMN 2.0 and in relation to soundness.

The hypothesis of this work is: *It is possible to generate business process-oriented text from natural language text*. From the defined hypothesis, the following specific goals are defined:

- Define how a business process-oriented text should be.

- Define an architecture that allows the generation of business process-oriented text.
- Check the similarity of the business process-oriented text to the original process.

To achieve these goals, existent techniques for discovering processes from natural language texts and techniques for generating natural language texts capable of describing business processes are taken into account in this work.

1.3 Contributions

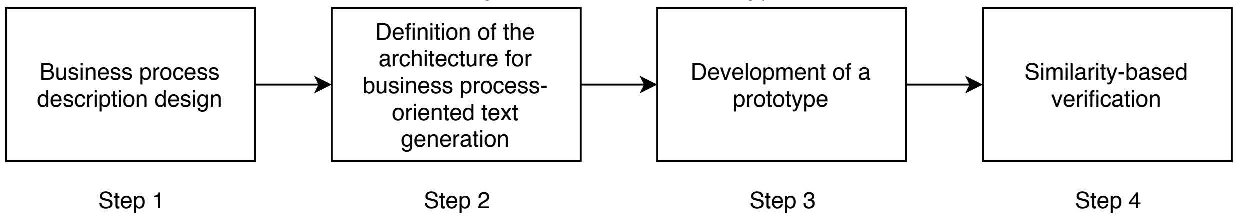
The main contributions of this dissertation, which are detailed in the conclusions, are:

- **Process Description Design:** In order to define how a process description should be described, an analysis was carried out on 64 process descriptions. This analysis consisted of a structural analysis of the texts and an analysis of sentence templates. Further, the analysis allowed to identify which sentence templates appear recurrently in the literature even presenting some kind of ambiguity issue. The results of this analysis may contribute to other approaches that describe business processes.
- **Architecture for generation of business process-oriented text:** In order to produce a process-oriented text, a service-oriented architecture consisting of five services was defined. Each service performs a certain functionality in order to produce the process description. In addition, contracts for communication between these services were deferred.
- **Prototype:** A prototype has been implemented for a generation of the process-oriented text. This prototype was constructed considering the defined approach for generation of process-oriented text. In addition, this prototype seeks to produce process descriptions that meet an analysis performed in the process description design.

1.4 Methodology

For the generation and validation of the business process-oriented text, this work presented a methodology that consists of four steps. Figure 1.2 presents the four steps of the methodology and how they are related.

Figure 1.2: Methodology.


Business process description design
Definition of the architecture for business process-oriented text generation
Development of a prototype
Similarity-based verification
Step 1
Step 2
Step 3
Step 4

Source: The authors

First, it was defined through empirical analyzes how the business process-oriented text should be (i.e., Step 1). Next, the architecture for business process-oriented text generation was defined (i.e., Step 2). Then, to enable testing of the created architecture, a prototype was developed (i.e., Step 3). Finally, a verification technique based on similarity was used to verify if the text generated by the approach is in accordance with its respective process (i.e., Step 4).

1.5 Remainder

This dissertation is divided into six chapters, considering this introduction. The other chapters are organized as follows:

- Chapter 2 outlines the fundamentals of Business Process Management (BPM), process modeling languages, Natural Language Processing (NLP) and Service Oriented Architecture (SOA). Initially, the definition of BPM and the description of its life cycle is presented. Regarding process modeling languages, this chapter explores the two languages used in this work, being: BPMN and Yet Another Workflow Language (YAWL). Then, it is defined NLP and presented its main concepts that will be used by this work. In addition, the concepts of SOA are described. Finally, the related works of this dissertation are presented.
- Chapter 3 presents the analysis performed for the definition of the design of business processes descriptions. First, this analysis takes into consideration how business processes descriptions are usually structured. Next, these process descriptions are analyzed with the objective of identifying recurrent sentence templates that do not present ambiguity issues.
- Chapter 4 presents the architecture to the generation of business process-oriented texts. In this chapter, the services created and the contracts defined for the interac-

tion between them are presented.

- Chapter 5 presents the developed prototype. In addition, this section reports the results of the similarity check between the business process-oriented text and the original process, and presents an analysis of the results.
- Chapter 6 presents the final conclusions of this work, highlighting the main contributions, limitations and possible future works.

2 FUNDAMENTALS

This chapter presents the main concepts that underlie this work. Section 2.1 introduces BPM and the business process lifecycle. Section 2.2 presents the process modeling languages BPMN 2.0 and YAWL. Section 2.3 defines NLP and describes the main concepts and techniques that were used in this dissertation. Section 2.4 defines SOA and presents Representational State Transfer (REST) as a way to obtain a SOA. Section 2.5 presents the related works of this dissertation. Finally, Section 2.6 presents the final considerations of this chapter.

2.1 Business Process Management

BPM is a discipline that involves concepts, methods, techniques, and tools for discovering, analyzing, redesigning, executing, and monitoring business processes (DUMAS et al., 2018). BPM can be seen from the perspective of a lifecycle formed by a set of steps that have well-defined objectives and are directly related.

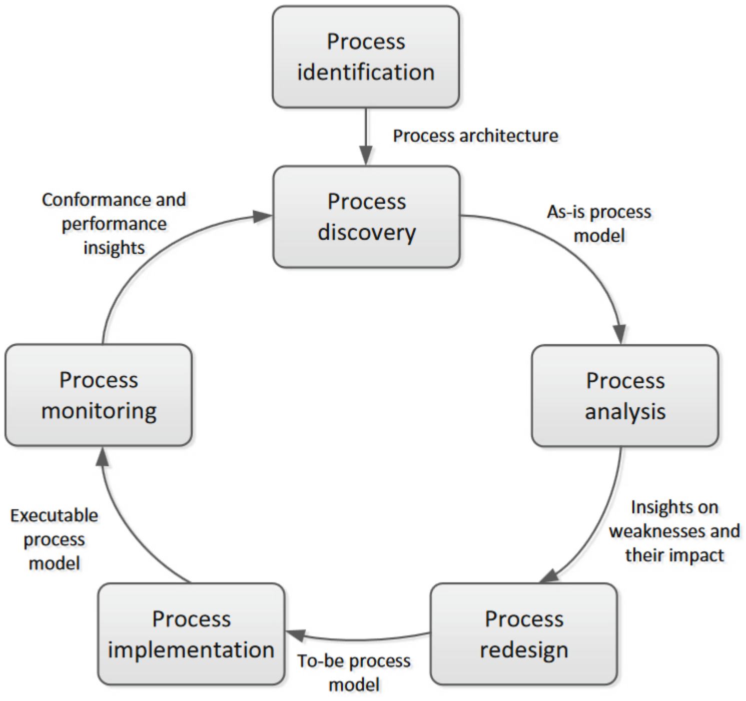
Several *BPM lifecycles* have been proposed by different authors in books and papers, such as: Aalst (2004), Aalst (2013), Weske (2012), Muehlen and Ho (2005), Brocke, Rosemann et al. (2010), Hallerbach, Bauer and Reichert (2008). In order to describe BPM phases, this work chose to present the lifecycle proposed by Dumas et al. (2018) because it is a very complete and easy to understand lifecycle, covering different BPM levels of abstractions in its execution. Figure 2.1 presents the BPM lifecycle proposed by Dumas et al. (2018). This lifecycle is composed by 6 phases, being:

- **Process identification:** In this phase, two goals are expected. Firstly, it is defined which business problems will be considered by the analysis. Secondly, the processes relevant to these business problems are identified, delimited, and related. This phase results in a process architecture that provides an overview of the processes existent in an organization and its relationships.

- **Process discovery:** In this phase, the relevant organizational processes defined earlier are understood and modeled in the way they currently occur. This phase is also known as "*as-is* process modeling". The result of this step is a set of *as-is* process models.

- **Process analysis:** In this phase, qualitative and quantitative analyses of what can be

Figure 2.1: BPM lifecycle.



Source: Dumas et al. (2018)

improved in the *as-is* models are carried out, taking into account a set of objectives and metrics to be achieved.

- **Process redesign:** In this phase, the creation of new processes based on the observations made in the process analysis phase occurs. This phase is also known as "process improvement". The result of this step is a set of to-be process models.

- **Process implementation:** In this phase, the changes necessary to transform the *as-is* process into the to-be process are prepared and executed. This step involves modifying the organization to perform the new process (i.e., organizational change management) and automating the process through the use of IT systems such as a Business Process Management System (BPMS) (i.e, a system that makes feasible process automation).

- **Process monitoring:** In this phase, the monitoring of the new process to identify nonconformities and degradation problems is carried out. Once the process presents problems or is not achieving the expected results, a new life cycle iteration becomes necessary.

As this work aims to help communication between process analysts and domain experts in order to understand the processes of the organization as they occur, it is possible to point out that this work focuses on bringing contributions mainly to the *process discovery phase*.

2.2 Process Modeling Languages

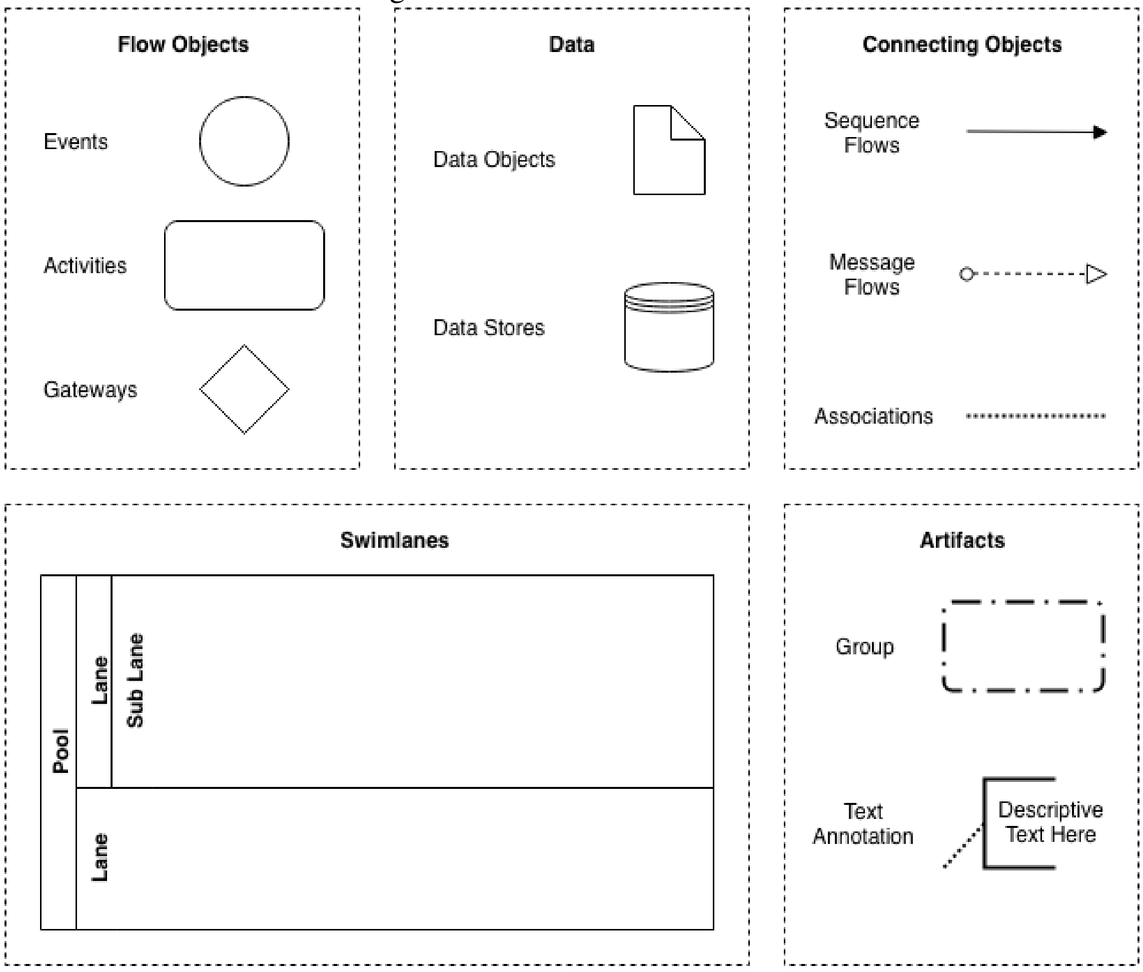
In this work, the BPMN 2.0 is used as the main language of the approach. Also, the soundness of a business process is supported by YAWL. Therefore, the remainder of this section introduces the fundamentals of BPMN 2.0 and YAWL.

2.2.1 Business Process Model and Notation

The BPMN 2.0 is a standard for process modeling maintained by the Object Management Group (OMG) (OMG, 2013). BPMN 2.0 includes five elements categories: *flow objects* (activities, events, and gateways), *data* (data objects, data inputs, data outputs, and data stores), *connecting objects* (sequence flows, message flows, associations, and data associations), *swimlanes* (pools and lanes) and *artifacts* (groups and text annotations). In this work, the focus is on *flow objects*, *swimlanes* and *connecting objects* because they are used for the identification and generation of business process descriptions.

Regarding *flow objects*, *activities* can be defined as a task that a company performs in a process (e.g., “create a document”, “make a purchase”). An *activity* can be atomic or non-atomic and is represented as rounded boxes. *Events* are described as some things that happen in the process and usually have a cause (e.g., “boarding pass received”, “purchase order received”) or an impact (e.g., ”order rejected”, “order fulfilled”). *Events* are represented as circles and indicate where a particular process starts (*start event*) or ends (*end event*). Moreover, there are *events* that can occur between a *start event* and an *end event* (*intermediate event*) which can affect the flow of the process but cannot start it or end it. Finally, *gateways* are represented as a diamond shape and are responsible for controlling divergence (split) and convergence (join) of *sequence flows* in a process. Thus, a gateway can lead to different paths (split) or join different paths into one (join). A path, in turn, can be defined as a set of *flow objects* connected sequentially through sequence flows. There are six different types of *gateways* which differ in both the logic that they execute and the representation placed within the gateway diamond. Among them, it can be highlighted: *exclusive gateway* (XOR, represented with or without a “X” marker), where the decision

Figure 2.2: BPMN 2.0 elements.



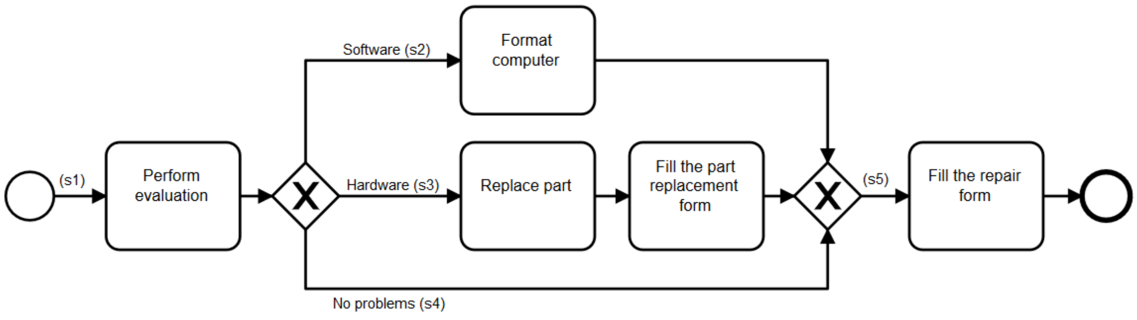
Source: The authors, adapted from Weske (2012)

making leads to the execution of exactly one path (e.g., "the payment must be made either with cash or with debit card"); *parallel gateway* (AND, represented with a "+" marker), where all possible paths must be executed (e.g., "the cook must prepare the food and the drink"); and *inclusive gateway* (OR, represented with a "O" marker), where decision making leads to the execution of at least one path (e.g., "the employee must request the materials from supplier 1 or supplier 2").

In relation to *swimlane*, a *pool* represents a participant in a business process. A *pool* is graphically represented as a container that partitions a process from other participants. If a *pool* does not contain a process, it is considered as a black box. *Lane*, on the other hand, are the partitions used to organize and categorize *activities* within a *pool*. *Lanes* are often used for representing internal roles (e.g., "manager", "clerk"), systems (e.g., "an enterprise application"), or an internal department (e.g., "shipping", "finance").

Regarding *connecting objects*, *sequence flows* are used to show the order of *flow objects* in a process. A *sequence flow* is represented as a solid single line with a solid

Figure 2.3: Example of BPMN model: Computer repair.



Source: The authors

arrowhead. A *message flow* represents the flow of messages between two different participants and is represented as a dashed single line with an open circle line start and an open arrowhead line end. In addition, an *association* is used to link information and artifacts with flow objects. *Data associations*, on the other hand, are used to relate data objects and *activities*. Both *association* and *data association* are represented as a dotted single line.

Figure 2.3 presents an example of a BPMN 2.0 process model composed by one start event, five activities, one exclusive decision gateway (XOR-split), one exclusive merge gateway (XOR-join), and one end event. After the process starts, an activity is executed (called “Make evaluation”). Then, there is a decision making in which only one of three possible paths can be followed. After one path is followed, the process returns to the main path, another activity is performed and the process ends. A possible description of the process shown in Figure 2.3 can be seen in Figure 2.4. The relationships between the text and the model are evidenced through s_i, where i refers to the sentence number in the text.

Figure 2.4: Example of business process description: Computer repair.

(s_1) When the process starts, the technician must *perform an evaluation in the computer*. (s_2) If there is a software problem, the technician must *format the computer*. (s_3) If there is a hardware problem, the technician must *replace the part* and *fill out the part replacement form*. (s_4) On the other hand, if no problem is found, *no modification should be made to the computer*. (s_5) The process finishes after the technician *completes the repair form*.

Source: The authors

BPMN 2.0 business process models can be represented and stored in files in the BPMN format. This file format is a standard maintained by OMG to represent processes following BPMN 2.0 (OMG, 2013).

2.2.2 Yet Another Workflow Language

YAWL (AALST; HOFSTEDE, 2005) is a workflow language that took as its starting point Petri Nets and was extended with constructs to address some workflow patterns (AALST et al., 2003; RUSSELL et al., 2006), such as: multiple instances, advanced synchronisation, and cancellation.

In addition, YAWL has an open source tool that performs syntactic verification and allows analysis of the model's soundness. The syntactic verification can be understood as checks necessary for the YAWL model to be considered correct. The analysis of soundness, in turn, seeks to identify execution problems in a workflow, even if it is considered to be syntactically correct. In this context, a workflow is considered with sound if, and only if, has the option to complete, has absense of dead tasks (e.g. unreachable parts), and has proper completion.

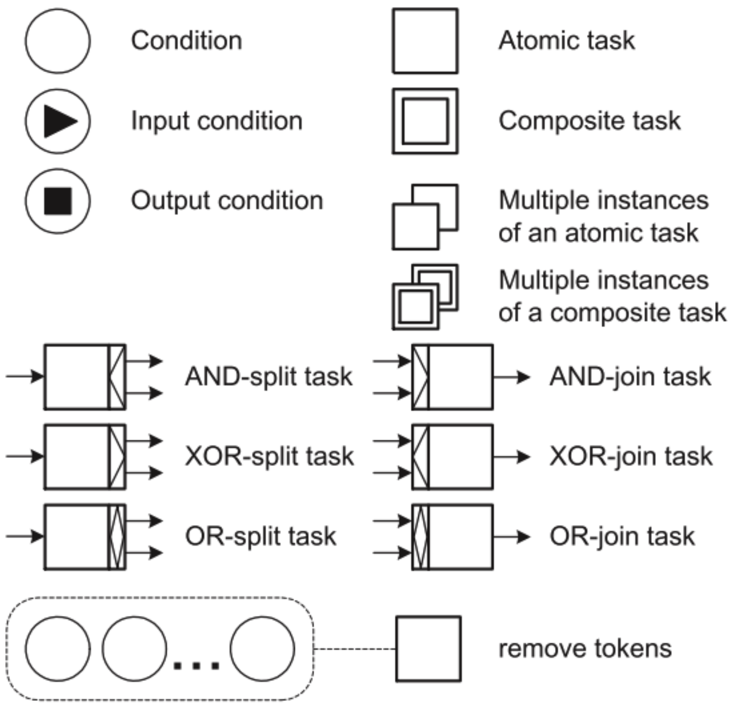
Figure 2.5 shows the symbols used in YAWL. The notation is composed by conditions and tasks linked by arrows. Conditions can be interpreted as places in the Petri Net. In the input condition, the execution tokens are created and in an output condition, the tokens are destroyed. In an YAWL model, there is exactly one unique input condition and one unique output condition. In contrast to Petri nets, it is possible to connect tasks (transition-like objects) without a condition (place-like object) between them.

Tasks represent work units to be performed. In the context of YAWL, there are atomic tasks and composite tasks. The difference between these is that the composite task represents a container for a sub-net with its own set of YAWL elements. Moreover, both atomic tasks and composite tasks can have multiple instances created after the task is started. In addition, a threshold can be set, and by the time it is reached, all running instances are terminated and the task completes.

Furthermore, tasks can be attached with up to one split and one join. The YAWL allows three types of splits and joins to: define parallel paths (AND-split task and AND-join task), define exclusive paths (XOR-split task and XOR-join task), to define inclusive paths (OR-split task and OR-join task).

Finally, there is a notation to remove tokens. According to the syntax of this symbol, at the time the external task executes, all tokens belonging to the dashed rounded rectangle area bound to the task by the dashed line are removed. This notation can be

Figure 2.5: Symbols used in YAWL.



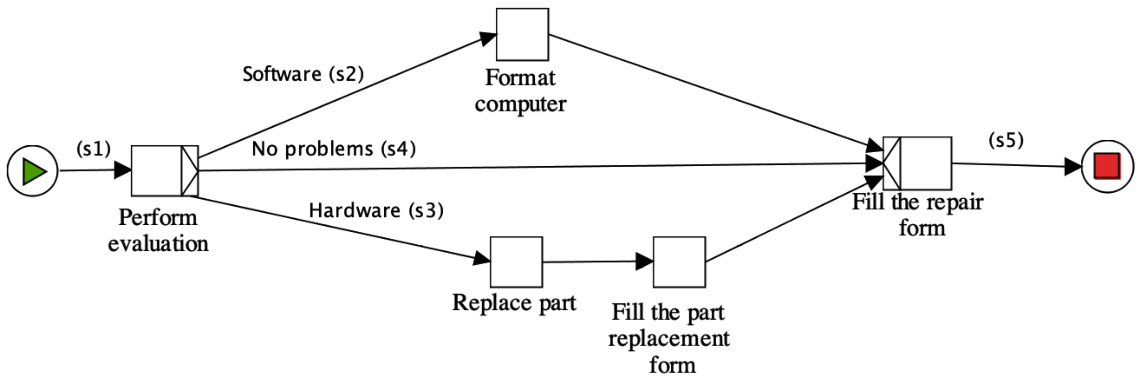
Source: Aalst and Hofstede (2005)

used to deal with cancellation in the workflow process.

Figure 2.6 presents an example of a YAWL process model based on the BPMN 2.0 model depicted in the Figure 2.3. This YAWL process model is composed by one input condition, five atomic tasks (i.e. being one a XOR-split task and one a XOR-join task) and one output condition. After the process starts, a task is executed (called “Make evaluation”). Then, there is a decision making in which only one of three possible paths can be followed. After one path is followed, the process returns to the main path, another task is performed and the process ends. As well as for the BPMN 2.0 process model example (Figure 2.3), the relationships between the business process description depicted in Figure 2.4 and the YAWL model are evidenced through s_x, where x refers to the sentence number in the text.

The YAWL is used in this work to ensure that the business process-oriented text is able to check the quality of the process in relation to soundness. In addition, YAWL’s choice lies in the fact that it has an implementation that is open source that prevents the approach being limited by a license and allows to make customizations to fit the approach developed in this work.

Figure 2.6: Example of YAWL model: Computer repair.



Source: The authors

2.3 Natural Language Processing

According to Jurafsky and Martin (2009), the goal of NLP is to get computers to perform useful tasks involving human language. These tasks may be to allow human-machine communication, to improve human-human communication, or doing useful processing of text or speech.

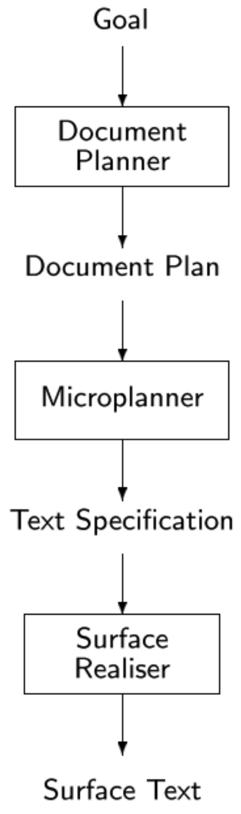
Reiter and Dale (2000) consider that the NLP is composed of two sub-fields, being: Natural Language Understanding (NLU) and Natural Language Generation (NLG). Taking into account this work is directly related to the creation of business process descriptions, the remainder of this section will introduce NLG focusing on its definition, applications and architecture.

Reiter and Dale (2000) define NLG as a sub-field of artificial intelligence and computational linguistics that is concerned with the construction of computer systems that can produce understandable texts in some human language generally starting from some non-linguistic representation of information as input.

One of the main uses of NLG is related to create computer systems that present information in a representation that people consider easy to understand (REITER; DALE, 1997). Among the applications of NLG, it is possible to mention: generating natural language texts from business process models (LEOPOLD; MENDLING; POLYVYANYY, 2012), generating natural language descriptions of object-oriented data models (LAVOIE; RAMBOW; REITER, 1996), generating natural language specifications from UML class diagrams (MEZIANE; ATHANASAKIS; ANANIADOU, 2008), generation of weather forecast descriptions (GOLDBERG; DRIEDGER; KITTREDGE, 1994), generation of bilingual (i.e. english and french) statistical reports (IORDANSKAJA et al., 1992).

According to Reiter and Dale (2000), NLG systems generally follow a three-stage

Figure 2.7: An NLG system architecture



Source: Reiter and Dale (2000)

pipeline architecture, composed by the following modules: document planner, microplanner and surface realiser. Figure 2.7 presents this NLG architecture.

In the document planner module, the activities of conceptual lexicalisation, content determination and document structuring are carried out. In the conceptual lexicalisation messages are constructed from the data input. In the content determination, it is decided which messages need to be communicated. Finally, in the document structuring is carried out the organization of these messages in order to generate a coherent and fluent text. The output of the document planner module is a document plan. A document plan is represented as a tree that specifies how the messages should be grouped and related to satisfy the initial communicative goal.

In the microplanner module, the activities of expressive lexicalisation, linguistic aggregation and referring expression generation are carried out. In the expressive lexicalisation is chosen which lexical items should be used to realise the conceptual elements in the messages. In the linguistic aggregation it is defined whether and how the messages should be combined. Finally, in referring expression generation is defined how the entities in the messages should be referred to (e.g. transform subjects that appear sequentially multiple times in pronouns). The output of the microplanner module is a text specification. A text specification is a data object that provides a complete specification of how the text should be generated. However, the natural language text is only possible after the accomplishment of the surface realiser module.

Finally, in surface realiser module, the text specification produced by the microplanner is transformed into text. At this stage, the message structures are converted into grammatically correct sentences.

2.4 Service Oriented Architecture

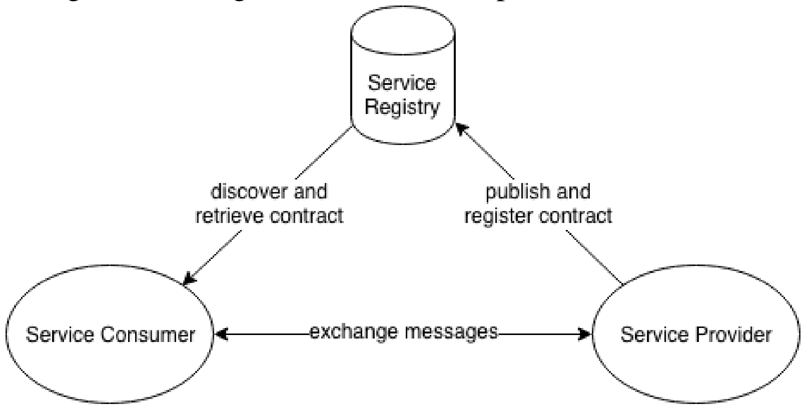
Acordling to Channabasavaiah, Holley and Tuggle (2003), SOA is: “an application architecture within which all functions are defined as independent services with well-defined invokable interfaces, which can be called in defined sequences to form business processes”. The service, on the other hand, is a software component provided through an endpoint accessible by the network, which has well-defined communication interfaces and supports the achievement of strategic objectives associated with service-oriented computing (ERL, 2008; PAUTASSO; ZIMMERMANN; LEYMANN, 2008).

The development of a SOA-based computing solution may be accompanied by many advantages. First, SOA facilitates modularization since services are handled separately and not as a single monolithic architecture. In addition, SOA is technology independent. Thus, it is possible that different services interact with each other even if they are implemented in different programming languages, software versions or operational systems. Moreover, SOA allows federation. In this context, services can work together, but each has its own autonomy and self-management.

Figure 2.8 presents the original architectural representation of SOA (ERL, 2008). This architecture consists of 3 different elements that represent roles that the services can play. These elements are:

- **Service registry:** Service that is a central element for the service discovery. This service is used at two different moments. At first, it is possible that existing services can be registered in a service registry reporting their location and the functionality they can provide. In the second moment, other services can access the service registry to make queries looking for a service that offers a certain functionality.
- **Service provider:** Service that is available to provide certain functionality. You can register for a service registry by entering the location of the service (i.e. the

Figure 2.8: Original architectural representation of SOA.



Source: The authors adapted from Erl (2008)

address of the service) and how to interact with it (i.e. the functionality it provides and how to access them).

- **Service consumer:** Service that consumes a particular functionality provided by another service. The service consumer searches the service registry for some service that provides some expected functionality. When the service registry finds a service that meets the service consumer request, the service registry sends data about the service to the service consumer so that the service consumer can, thus, communicate directly with the service provider and obtain the desired functionality.

A service can act as more than one role at different moments. Thus, it is possible for a service to sometimes act as a service provider, providing a service, and sometimes acting as a service consumer, making use of other services. Usually, services are expected to interact through well-defined interfaces called service contracts.

According to Erl (2008), a service contract establishes the terms of agreement between the service consumer and the service owner. The contract shall provide the technical constraints and requirements as well as any semantic information that the service owner wishes to make public. Contract consists of service description documents, such as: service definition files (e.g. WSDL, WADL), data definition files (e.g. XSD), and policy files (e.g. WS-Policy). In addition, contracts may also contain non-technical service documents, such as Service Level Agreement (SLA) documents.

According to Ren and Lyytinen (2008), SOA can be implemented in many ways, such as: Web services, Common Object Request Broker Architecture (CORBA), Distributed Component Object Model (DCOM). The most common is through web services (ERL, 2008; REN; LYYTINEN, 2008). Web services, in turn, can be of different approaches, such as: Simple Object Access Protocol (SOAP) and REST.

REST is an architectural style for distributed hypermedia systems (FIELDING; TAYLOR, 2000). REST is composed of a set of principles and rules for the construction of interoperable web services. In addition, REST performs communication through the hypertext transfer protocol (HTTP) and uses the verbs of this protocol (e.g. PUT, GET, POST, DELETE) to carried out communication between the different services.

Fielding and Taylor (2000) defines that REST follows the client-server architectural style. Furthermore, all communication is stateless and caching can be used to improve the network efficiency. Moreover, REST promotes a uniform interface between components, makes use of layered system (i.e. each component can see only the immediate layer with which they are interacting) and allows client functionality to be extended by downloading and executing code in the form of applets or scripts (i.e. code-on-demand).

Some of key benefits of REST when compared to SOAP are its simplicity and the recommendation of its use for ad-hoc integration over the Web (PAUTASSO; ZIMMERMANN; LEYMANN, 2008). In addition, REST has a low degree of coupling (PAUTASSO; WILDE, 2009), and is quite efficient in terms of performance (KUMARI; RATH, 2015). In addition, SOAP displays great verbosity in your messages (ZIMMERMANN et al., 2004). Considering that we want to develop an approach efficient in terms of performance, that allows communication between services with little effort (low verbosity in messages), and where new services can be easily added or replaced in future versions (low degree of coupling), this work opted for the use of REST for the implementation of SOA.

Although in the early SOA follow the architecture represented in Figure 2.8, it has undergone changes and modifications to suit the most diverse scenarios and technologies. As an example, Pautasso and Wilde (2009) points out that REST Web services are usually not described or registered in any standardized or centralized way. This idea goes against the goal of maintaining a centralized service discovery point (i.e. service registry) proposed by the original SOA architecture.

Still related to the discovery of services, in the first version of web services it was proposed the use of Universal Description, Discovery and Integration (UDDI) to act as a service registry for SOAP-based services. However, the UDDI were not widely adopted (ERL, 2008; PAUTASSO; ZIMMERMANN; LEYMANN, 2008). Therefore, some organizations end up accessing the services directly (i.e. the service consumer address is inserted into the service provider). Others, in turn, choose to implement their own way of discovering services (PAUTASSO; WILDE, 2009; ZIMMERMANN et al., 2004; BAL-

ANI; HATHI, 2009).

2.5 Related Works

The related works of this dissertation involve the application of NLP in BPM. Several recent studies have already carried out systematic literature review of works that relate BPM and NLP (BORDIGNON et al., 2018; RIEFER; TERNIS; THALER, 2016; MAQBOOL et al., 2018).

In this work, the related works were divided into three different streams of research, being: discovery of processes from natural language texts, generation of business process descriptions, and alignment between process model and business process description. In the following sections these works will be presented and discussed.

2.5.1 Discovery of Processes from Natural Language Texts

The works included in this category are related to the identification of business processes from texts. These works are relevant as they contribute to the process discovery stage helping to understand how the organization's processes are documented. For this category, twelve works were found and divided into three different groups: *generate process models from texts*, *process mining from natural language text* and *activity label*. While the first two groups act on the identification of the whole process, the third group, in turn, act directly on the scope of the activity label. Table 2.1 presents the works of this category.

Vakulenko (2011) proposes a method for semi-automated process model extraction from business process descriptions. The method was designed according to the specific patterns and vocabulary typical for documents containing business process specifications. The method presented high precision and recall for activity extraction. However, it presented lower results when trying to extract the relationship between the process elements. One of the limitations pointed out by the author was that the tool was not able to detect and resolve anaphoras (e.g., to understand a particular actor or object that is represented in the text as a pronoun or determinant). Differently from the work presented in this dissertation, the work proposed by Vakulenko (2011) aims to produce a process model and not a business process description.

Table 2.1: Related works 1

Discovery of Processes from Natural Language Texts
• Generate Process Models From Text
– (VAKULENKO, 2011)
– (FRIEDRICH; MENDLING; PUHLMANN, 2011)
– (FERREIRA; THOM; FANTINATO, 2017)
– (FERREIRA et al., 2017)
– (CAPORALE, 2016)
– (GHOSE; KOLIADIS; CHUENG, 2007)
– (HEUSER; ELSTERMANN, 2016)
– (ELSTERMANN; HEUSER, 2016)
• Process Mining from Natural Language Text
– (GONCALVES; SANTORO; BAIAO, 2009)
– (GONÇALVES; SANTORO; BAIÃO, 2010)
– (EPURE et al., 2015)
– (LI et al., 2010)
• Activity Label
– (PITTKE et al., 2015)
– (LEOPOLD; SMIRNOV; MENDLING, 2010)

Source: The authors

Friedrich, Mendling and Puhlmann (2011) proposes an approach that is able to identify process elements in texts, as well as their respective relationships, in order to produce a BPMN process model. In addition, the approach has its own technique of anaphora resolution that helps in identifying elements that are presented in the text as pronouns and determinants. The evaluation of this approach shows that for a set of 47 text-model pairs from industry and textbooks, the authors are able to generate on average 76.98% of the models correctly. Similar to the previous work, the work proposed by Friedrich, Mendling and Puhlmann (2011) aims to produce a process model and not a business process description. In any case, considering the characteristics of this work, the technique proposed by Friedrich, Mendling and Puhlmann (2011) to identifying the process elements and their respective relationships was adapted to be used by our approach to generate business process-oriented texts.

Ferreira, Thom and Fantinato (2017) proposes a semiautomatic approach that allows the identification and marking of process elements from a set of rules defined through an empirical analysis of business processes descriptions. This approach was validated through a survey that collected opinions about mapping rules and through precision, recall, f-measure and accuracy measurements. In addition, Ferreira et al. (2017) conducted a survey with the objective to evaluate the difference of modeling process when the modelers use rule-mapped text compared to the modeling done by the same modelers without this additional aid. This analysis focused mainly on measuring the time taken to complete the modeling task and the necessary effort perceived by the modeler. The approach presented in this dissertation also performs the marking of process elements in a business process description. However, these works are different because they identify the process elements in an original text, while our approach aims to generate a new business process description from the original text.

Caporale (2016) suggested a method that allows generating process models from business process descriptions. To achieve this, the author proposed that the business process descriptions should be specified with a controlled natural language, based on sentence templates, in order to facilitate the extraction of information necessary to generate the models. Similar to the other works in this section, the work proposed by Caporale (2016) aims to generate process models and not the generation of business process descriptions.

Ghose, Koliadis and Chueng (2007) proposed a framework and prototype tool that can query information resources (e.g., corporate documentation, web-content, code) for construct models to be incrementally adjusted to correctness by an analyst. One of the techniques used by authors to extract information from text is based on template extraction. In this technique, the authors created templates from textual structures that are commonly used in describing processes and used these templates to extract knowledge from text documents. Similar to the other works in this section, the work proposed by Ghose, Koliadis and Chueng (2007) aims to build process models and not the generation of business process descriptions.

In addition to the works cited above, there were works related to the discovery of processes from descriptions made by domain experts (HEUSER; ELSTERMANN, 2016; ELSTERMANN; HEUSER, 2016), technique of model generation through mining of group stories (GONCALVES; SANTORO; BAIAO, 2009; GONÇALVES; SANTORO; BAIÃO, 2010), technique of mining activities from texts (EPURE et al., 2015) and automatic extraction of process information from policy documents through the use of machine learning techniques (LI et al., 2010). Moreover, there were works related to the definition of activity labels (PITTKE et al., 2015) and the automated activity label refactoring (LEOPOLD; SMIRNOV; MENDLING, 2010).

Although the work presented in this dissertation makes use of techniques for *discovery of processes from natural language texts*, the works identified in this stream of research are only intended to extract knowledge of business process descriptions or the generation of process models from business process descriptions. Thus, this work differs from the works present in this section because it seeks, based on information extracted from a business process description, produce a business process-oriented text.

2.5.2 Generation of Business Process Descriptions

The works included in this category are related to the generation of business process descriptions from process models. These works are relevant since they may provide textual documentation capable of accompanying a process described by a process modeling language. Among the main applications is to help people with no knowledge in the process modeling language understand the process. For this category, seven works were found and divided into two different groups: *generation of business process descriptions from process models* and *text structuring*. Table 2.2 presents the works of this category.

Table 2.2: Related works 2

Generation of Process Descriptions
• Generation of Process Descriptions from Process Models – (LEOPOLD; MENDLING; POLYVYANYY, 2012) – (LEOPOLD; MENDLING; POLYVYANYY, 2014) – (MEITZ; LEOPOLD; MENDLING, 2013) – (AYSOLMAZ et al., 2018) – (RODRIGUES; AZEVEDO; REVOREDO, 2016) – (MALIK; BAJWA, 2012) • Text Structuring – (FERRARI et al., 2017)

Source: The authors

Leopold, Mendling and Polyvyanyy (2012) propose an approach that allows the generation of natural language text from a BPMN process model. This approach is composed of three stages of natural language generation, being: *sentence planning*, *text planning* and *realization*. One of the advantages of this approach is that it deals with structures that are recurrently encountered in business processes (e.g., a XOR gateway leading to two activities), but also presents a a graph-based way to describe parts of the process that do not fit these structures. In addition, Leopold, Mendling and Polyvyanyy (2014) extend the approach by conducting a validation with users in which they ask users to translate the generated texts back into the process models in order to investigate whether humans can successfully understand the generated texts. The approach proposed in these works was adapted to be used by our approach of generation of business process-oriented text. Among the changes, the approach was modified to have sentence templates that were recurrently found in actual business process descriptions. In addition, some sentence templates have been modified in order to reduce ambiguity issues in the business process descriptions created.

Meitz, Leopold and Mendling (2013), in turn, propose a natural-language text generation approach from Petri nets to help people with little modeling experience understand and validate their process models. The main difference from our approach is that our approach produces text from business process description while this approach produces text from a Petri net.

Aysolmaz et al. (2018) defined a semi-automated approach to generate natural language requirements documents (i.e., list containing a sequence of steps to be performed) based on business process models. The authors adopted a template filling technique, in which sentence templates are defined containing gaps that must be filled with information from a requirements model. The main difference from our approach is that Aysolmaz et al. (2018) generates a requirement document, while our approach produces a textual description.

Rodrigues, Azevedo and Revoredo (2016) propose a language-independent framework that automatically generates natural language texts from BPMN business process models. The authors propose a framework that works for two languages, being: English and Portuguese. However, the authors comment that the approach was conceived as an independent language solution, which can be applied theoretically to any language. Similar to other works already cited in this section, the work presented by Rodrigues, Azevedo and Revoredo (2016) has as input a BPMN process model, while our approach receives as input a business process description.

Malik and Bajwa (2012) present a novel approach to automatically generate natural language representation of business process models explained in BPMN. The presented approach employs Semantics of Business Vocabulary and Rules (SBVR) as an intermediate representation to generate natural language expressions those are easy to understand for business stakeholders. Among the limitations, the text produced by this approach does not perform some common NLG techniques in order to make the text more readable (e.g., performing *referring expression* in order to transform subjects that appear sequentially multiple times in pronouns). In addition, the approach deals with recurring structures in business processes but, unlike Leopold, Mendling and Polyvyanyy (2012), does not make it clear how to deal with parts of the process that do not follow one of these structures.

Ferrari et al. (2017) conducted a literature review and a set of interviews with different public institutions aiming at improving the business process descriptions to be used in public administrations. Among the main contributions, the authors provide a set of guidelines to guide the construction of business process descriptions. In addition, the authors concluded that there are four macro-areas of research in which computer scientists can contribute towards more quality in business process descriptions, being: *readability*; *ambiguity*; *relevance and text summarisation*; *modelling and consistency*. The main difference in relation to our approach is that our approach generates a business process description while the objective of Ferrari et al. (2017) is providing guidelines to guide the construction of business process descriptions.

Although the work presented in this dissertation makes use of techniques for the *generation of business process descriptions*, the works identified in this stream of research are only intended to define how the business process descriptions should be or to perform the automatic generation of business process descriptions from business process models. Thus, this work differs from the others presented in this section because it produces a business process-oriented text from a business process description. In addition, this work also performs a verification of the process described in relation to BPMN 2.0 and in relation to the soundness.

2.5.3 Alignment between Process Model and Business Process Description

The works included in this category are related to the alignment between process model and business process description. These works are relevant since they help to find inconsistencies between business process descriptions and process models that represent the same process. Among the main applications is to avoid nonconformities in the process documentation. For this category, three works were found and divided into two different groups: *inconsistencies between process models and business process description* and *ambiguity*. Table 2.3 presents the works of this category.

Table 2.3: Related works 3

Alignment between Process Models and Process Descriptions
• Inconsistencies between Process Models and Process Description – (AA; LEOPOLD; REIJERS, 2015) – (AA; LEOPOLD; REIJERS, 2018) • Ambiguity – (AA; LEOPOLD; REIJERS, 2016)

Source: The authors

Aa, Leopold and Reijers (2015) presented an approach to automatically detect inconsistencies between process model and the corresponding textual description. The authors have identified that a technique for detecting inconsistencies between a process model and a business process description must deal with the ambiguity present in the natural language. This must be done because an ambiguous sentence may produce multiple interpretations of a process (e.g., it is not possible to define with certainty whether in the sentence “Then, the passenger must *pass through the security screening* and the luggage must *pass through the luggage screening*.” the activities described in italics are executed in sequence or in parallel).

In another work, Aa, Leopold and Reijers (2016) proposed to deal with ambiguity in business process descriptions introducing the *behavioral space* concept. According to the authors, the *behavioral space* captures all possible behavioral interpretations of a business process description. Thus, a given business process description containing an ambiguity could generate multiple business process descriptions representing each possible interpretation.

Furthermore, Aa, Leopold and Reijers (2018) presented an approach to verify the compliance between a process model and a business process description, considering the ambiguity present in texts. To handle the ambiguity issues, they used the concept of *behavioral space* previously proposed by Aa, Leopold and Reijers (2016).

Differently from the work presented in this dissertation, the works identified in this stream of research are intended to find inconsistencies between business process descriptions and process model. Thus, this work differs from the works presented in this section because this work seeks to generate business process-oriented texts from natural language texts, while the works of this section aims to compare texts and process models.

2.6 Final Considerations

This chapter presents the fundamentals used in this dissertation, being them: BPM, BPMN 2.0, YAWL, NLP, and SOA.

In relation to BPM, the different phases of the BPM lifecycle for the identification, generation, and management of business processes in the organizations were presented. Moreover, it was discussed in which phase of the lifecycle this work is inserted.

In addition, this chapter presented the fundamentals of some process modeling languages used in this work. Firstly, it presented the main concepts of BPMN 2.0, as well as gave an overview of its notational elements and addressed the process elements used in the scope of this dissertation. Then, was presented the notation YAWL and discussed how the same can be applied for verification of soundness in a process.

Subsequently, this chapter presented NLP, with the main focus being the NLG sub-field. For NLG, the chapter presented its definition, applications, and a pipeline architecture of how NLG systems are generally created. Regarding this architecture, each module with its respective inputs and outputs has been presented.

Regarding SOA, it was presented its definition, its main advantages, and its original architectural representation. In addition, the chapter defines what services and contracts are from the perspective of SOA. Moreover, different ways of implementing SOA were presented, focusing on the architectural style REST.

In the context of related works, works were presented that involve the application of NLP in BPM. These works were divided into three different streams of research, being: *discovery of processes from natural language texts*, *generation of business process descriptions*, and *alignment between process model and business process description*. Among these three streams of research, this work is more related to the *generation of business process descriptions*. However, as this work aims to generate the business process-oriented text from a natural language text, it is also possible to relate the work presented in this dissertation to the *discovery of processes from natural language texts*.

Besides the main differences already presented between this work and the related works, this work presents other characteristics that distinguish it from the presented related works. Firstly, this work presents an empirical analysis in 64 business processes descriptions in order to understand how business process descriptions are usually written. This analysis includes understanding the structure of a business process description, find recurrent sentence templates used in a business process description, and to highlight ambiguity issues in these sentence templates. In addition, the business process descriptions generated by this work are intended to assist in understanding the process by previously identifying possible nonconformities with BPMN 2.0 and soundness. Finally, this work proposes the interaction with the user, since it can mark the process elements, highlight the activities carried out by certain resources, and perform the restructuring of the text. This restructuring can be done by defining the size of the paragraphs, indenting the text, and adding bullet points in sentences of the text.

3 BUSINESS PROCESS DESCRIPTION DESIGN

For the generation of a business process-oriented text, it was first necessary to define how this process description should be. For this, it was necessary to define how the text would be structured (e.g., paragraph size, use of active or passive voice, how certain parts of a process are represented in a business process description) and how the sentences that would compose the text should be. To achieve these objectives, in the context of the present work, empirical analyses were carried out on existing business process descriptions and literature works, in order to discover how business process descriptions are described. Then, the analyses were used to define how a business process-oriented text should be.

The first analysis that was carried out was intended to understand how process descriptions are usually structured. To this end, five questions were asked about how business processes are usually described. To answer these questions, we analyze business process descriptions found in the literature. In addition, we considered in this analysis works in the literature that analyze business process descriptions with the purpose of generating process models, and works that suggest how texts in natural language can be structured. The answer to these questions will help define how business process-oriented texts should be structured.

The second analysis aimed to understand how the sentences that make up the text usually are. For this, an analysis was carried out on business process descriptions found in the literature with the objective of identifying recurring sentence templates. In the literature, a number of approaches have been found that use sentence templates to generate business process descriptions or to transform process descriptions into process models. However, the corresponding approaches do not explain how the sentence templates that compose the process description were selected and this information is important, as it directly interferes with the quality of the text. Sentence templates not carefully selected may produce process descriptions with ambiguity issues that may not be understood by the process analysts and domain experts. In order to avoid producing process descriptions with ambiguity issues, an analysis was performed in order to identify ambiguity issues that are presented in sentence templates. In the context of this work, a sentence template is considered with ambiguity issues when it allows multiple interpretations of the process. Finally, the sentence templates identified were categorized and classified as having ambiguity issues or not. This second analysis, as well as its results obtained, were published by the authors in the 26th International Conference on Cooperative Information Systems - CoopIS 2018 (SILVA et al., 2018).

The remainder of this chapter is structured as follows. Section 3.1 introduces the business process descriptions used in the empirical analyses. Section 3.2 presents the analysis to understand how the business process descriptions are usually structured. Section 3.3 presents the analysis for the definition of the sentences that will compose the text. Section 3.4 describes how the process descriptions will be taking into account the information collected in the previous analyses. Section 3.5 provides an example of a process description created from the design decisions suggested in this chapter. Finally, Section 3.6 presents the final considerations of this chapter.

3.1 Business Process Descriptions

The process descriptions used in the analysis came from two different sources and only process descriptions in English were considered, as templates are very sensitive to the language.

Firstly, 47 process descriptions present in Friedrich (2010) were identified. From this first source, 17 process descriptions were disregarded for the following reasons: they were translated from another language through machine translation services (14 texts), were duplicated (2 texts) or had a description format based on enumeration (1 text).

Secondly, 34 process descriptions from the book Fundamentals of Business Process Management (DUMAS et al., 2013) were identified. The final set of 64 process descriptions, as well as their respective sources and types are presented in Table 3.1. In addition, the name of all business process descriptions as well as the repository for accessing them can be seen in the Appendix.

3.2 Analysis for Structuring the Text

In this section, we perform an analysis in business process descriptions and literature works with the objective of answering five questions. These questions have arisen from the need to understand how business process descriptions are usually structured. The five questions are:

- **Question 1:** How is the text usually described in relation to the process?

Table 3.1: Data sources.

Source	Amount	Type
HU Berlin	4	academic
TU Berlin	2	academic
QUT	8	academic
TU Eindhoven	1	academic
Vendor Tutorials	4	industry
inubit AG	3	industry
BPM Practicioners	1	industry
BPMN Prac. Handbook	3	textbook
BPMN M&R Guide	4	textbook
Fundamentals of BPM	34	textbook
Total	64	–

Source: The authors

- **Question 2:** How is the text organized in terms of paragraphs?
- **Question 3:** What is the voice used in the text?
- **Question 4:** How does the text describe splits and joins?
- **Question 5:** How does the text describe the different paths generated by splits?

In the following sections each of these questions will be detailed and possible answers will be presented.

3.2.1 Question 1: How is the text usually described in relation to the process?

This question is intended to understand how the business process is usually organized into a process description. In this sense, this question seeks to understand at what point the notational elements are represented in the process descriptions.

From the business process descriptions analyzed, it was observed that the texts describe the processes following the sequence of the execution of their activities. When compared to a process model represented in BPMN 2.0 notation, the process description begins by describing the *start events* and ends by describing the *end events*.

In addition, works that extract business process information from natural language texts usually consider that activities are sequentially described in the text, following a chronological order of when they are executed (SCHUMACHER; MINOR; SCHULTE-ZURHAUSEN, 2013; VAKULENKO, 2011; FRIEDRICH; MENDLING; PUHLMANN, 2011; FERREIRA; THOM; FANTINATO, 2017).

3.2.2 Question 2: How is the text organized in terms of paragraphs?

This question is intended to understand whether the process descriptions are structured using paragraphs. If they are, this question seeks to understand the causes that lead to the creation of paragraphs, for example: to reach a certain number of words, to reach a certain number of sentences, to describe some process element, to describe some common structure in processes. In order to find out the different reasons that lead to the creation of new paragraphs, we analyzed the business process descriptions and works existing in the literature that discuss the subject. It is important to emphasize that the purpose here is not to find the best way to divide the text into paragraphs, but rather the different possible ways so that one or more can be used in the design of the business process-oriented text.

This question was difficult to answer considering the process descriptions collected by data sources. The reason is that the process descriptions presented by Friedrich (2010) are all described as a single paragraph. However, when searching for the original sources (or, in some cases, in other sources that these texts appear), it has been observed that some of these process descriptions can also be found divided into multiple paragraphs. In this sense, we tried to find through internet searches others sources (e.g., books, examples provided by process modeling tools) in which these descriptions appeared. Of the 30 process descriptions considered, only 15 were identified. In relation to the identified texts, five presented multiple paragraphs, being: Underwriters (Pd_4), Claims Notification (Pd_{12}), Intaker Work (Pd_{15}), Oracle Tutorial (Pd_{19}), and Exercise 5 (Pd_{33}). These texts contained an average of 4.4 paragraphs and an average of 3.77 sentences per paragraph. Among the different reasons that led to the creation of new paragraphs, it is possible to highlight: to describe a new event, to describe the assignments of a resource, to describe different gateways (i.e., each gateway would be a new paragraph), to show the different paths of a gateway, and to join the paths of a gateway.

Regarding the process descriptions present in Dumas et al. (2013), it was identified that from 34 texts analyzed only 2 presented multiple paragraphs, being: Exercise 4.18 (see Figure 3.1) and Exercise 4.33. Both texts contained five paragraphs and an average of 4.4 sentences per paragraph. In addition, it was observed that some paragraphs contained context information (i.e., did not describe the process itself) or information that could occur during the entire process (e.g., “Any time during the process, the manager may require the report.”).

The definition of the use of paragraphs in the text is not limited to the generation of

Figure 3.1: Process description: Exercise 4.18.

The mortgage application process starts with the receipt of a mortgage application from a client. When an application is sent in by the client to the broker, the broker may either deal with the application themselves, if the amount of the mortgage loan is within the mandate the broker has been given by BestLoans, or forward the application to BestLoans. If the broker deals with the application themselves, this results in either a rejection or an approval letter being sent back to the client. If the broker sends an approval letter, then it forwards the details of this application to BestLoans so that from there on the client can interact directly with BestLoans for the sake of disbursing the loan. In this case, BestLoans registers the application and sends an acknowledgment to the client.
The broker can only handle a given number of clients at a time. If the broker is not able to reply within one week, the client must contact BestLoans directly. In this case, a reduction on the interest rate is applied should the application be approved.
If BestLoans deals with the application directly, its mortgage department checks the credit of the client with the Bureau of Credit Registration. Moreover, if the loan amount is more than 90% of the total cost of the house being purchased by the client, the mortgage depart- ment must request a mortgage insurance offer from the insurance department. After these interactions BestLoans either sends an approval letter or a rejection to the broker, which the broker then forwards to the client (this interaction may also happen directly between the mortgage department and the client if no broker is involved).
After an approval letter has been submitted to the client, the client may either accept or reject the offer by notifying this directly to the mortgage department. If the mortgage department receives an acceptance notification, it writes a deed and sends it to an external notary for signature. The notary sends a copy of the signed deed to the mortgage department. Next, the insurance department starts an insurance contract for the mortgage. Finally, the mortgage department submits a disbursement request to the financial department. When this request has been handled, the financial department notifies the client directly.
Any time during the application process, the client may inquire about the status of their application with the mortgage department or with the broker, depending on which entity is dealing with the client. Moreover, the client may request the cancellation of the application. In this case the mortgage department or the broker computes the application processing fees, which depend on how far the application process is, and communicates these to the client. The client may reply within two days with a cancellation confirmation, in which case the process is canceled, or with a cancellation withdrawal, in which case the process continues. If the process has to be canceled, BestLoans may need to first recall the loan (if the disbursement has been done), then annul the insurance contract (if an insurance contract has been drawn) and finally annul the deed (if a deed has been drawn).

Source: The authors adapted from Dumas et al. (2013)

business process descriptions. In terms of NLP, different works were found that present approaches for the automatic segmentation of natural language texts in multiple paragraphs (HEARST, 1994; HEARST, 1997; HEINONEN, 1998). These studies consider, among other things, factors such as the similarity between sentences and the expected size of paragraphs to define how the text will be divided into paragraphs.

3.2.3 Question 3: What is the voice used in the text?

This question is intended to understand whether the process descriptions are usually described using the active voice or the passive voice. An example of an active voice would be the sentence s_1 of Figure 2.4 (i.e., “When the process starts, the technician must *perform an evaluation in the computer.*”). This sentence could be written in the passive voice as “When the process starts, *an evaluation on the computer should be performed.*”. As can be seen in both sentences, the sentence representing the passive voice would not be possible to recognize who is the actor that performs the action of evaluating the computer. This happens because not always the passive voice will contain the information of who carries out an action.

The absence of information from the actor performing the action is a characteristic present in many sentences in the passive voice that is undesirable in process descriptions since without knowing who is the actor that performs an action, it is possible to create process descriptions where there are activities without explicitly knowing the resource responsible for performing them.

In the process descriptions analyzed it was observed that the majority of them describes the process in the active voice (51.57% of the business process descriptions), followed by business process descriptions described in the passive voice (32.81% of the business process descriptions). In addition, we found business process descriptions that use both the active voice and passive voice to describe the process (15.62% of the business process descriptions).

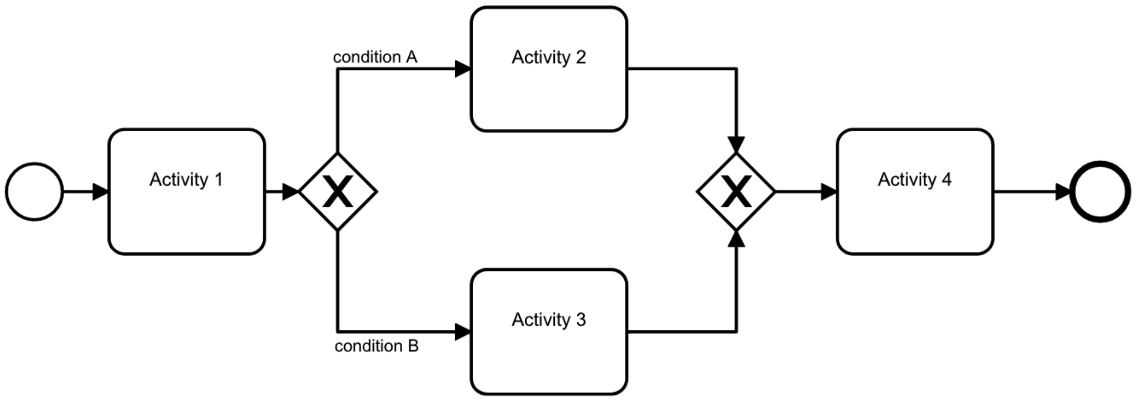
3.2.4 Question 4: How does the text describe splits and joins?

This question is intended to understand how process descriptions describe splits and joins. To answer this question, the business process descriptions presented in Section

3.1 have been analyzed. The different ways of representing splits and joins were called *cases*. Each *case* will be presented containing a specific notation, a description, and an example of process description.

For the notation, in the present work *cases* were defined as $Case_{4-ti}$ where 4 represents the question number, t represents its type (i.e., s for split and j for join), and i represents an index associated with a given case (index starts with the value of 1 and will be incremented to represent each possible new case). In order to complement the description, each case will present an example of process description highlighting, when possible, where in the process description the case appears. The process descriptions created for each case are intended to describe the BPMN 2.0 process model with generic activities presented in Figure 3.2. This process model was created because it is an easy example to understand the different cases related to this question.

Figure 3.2: Question 4: Process model example.



Source: The authors

From the business process descriptions analyzed, two ways of describing splits were derived, being them:

- $\mathbf{Case_{4-s1}}$: A sentence is used to describe splits.
  - Example: The process starts by performing activity 1. *Then one of the following paths can be performed.* If condition A is true, activity 2 must be done. However, if condition B is true, activity 3 must be done. In any case, the activity 4 is performed. Finally, the process ends.
- $\mathbf{Case_{4-s2}}$: A sentence is not used to describe splits. In this case, split is implied in markers, such as: if, else, in case that, for the case.
  - Example: The process starts by performing activity 1. If condition A is true,

activity 2 must be done. However, if condition B is true, activity 3 must be done. In any case, the activity 4 is performed. Finally, the process ends.

In the business process descriptions analyzed, three ways of describing joins were identified, being them:

- $\mathbf{Case_{4-j1}}$: A sentence is used to describe joins.
  - Example: The process starts by performing activity 1. If condition A is true, activity 2 must be done. However, if condition B is true, activity 3 must be done. *The process continues after one of the paths is executed.* Then, the activity 4 is performed. Finally, the process ends.
- $\mathbf{Case_{4-j2}}$: A sentence is not used to describe joins and these are implicit in the text.
  - Example: The process starts by performing activity 1. If condition A is true, activity 2 must be done. However, if condition B is true, activity 3 must be done. Activity 4 is then performed. Finally, the process ends.
- $\mathbf{Case_{4-j3}}$: A sentence is not used to describe joins, but the following sentence make use of discourse markers, such as: in any case, in all case, afterward.
  - Example: The process starts by performing activity 1. If condition A is true, activity 2 must be done. However, if condition B is true, activity 3 must be done. *In any case*, activity 4 is then performed. Finally, the process ends.

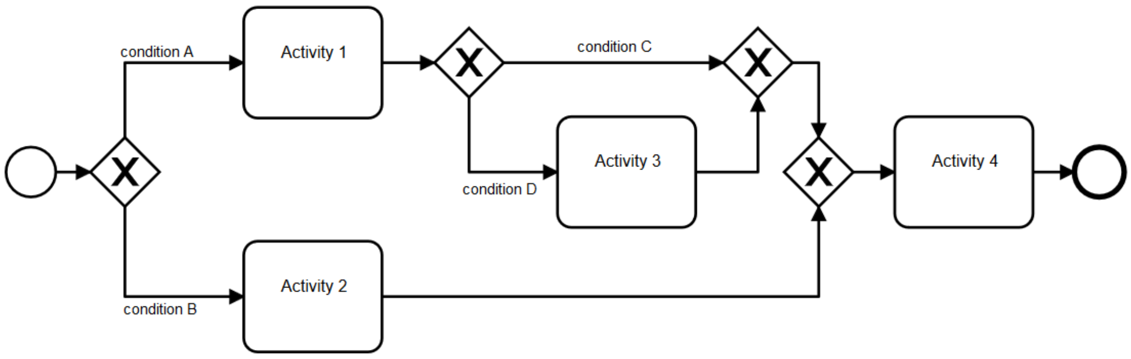
3.2.5 Question 5: How does the text describe the different paths generated by splits?

Regarding how the text behaves to describe the different paths generated by splits, the texts analyzed usually describe the paths in different ways. These ways are differentiated by some considerations, being: (1) when to stop describing a path (i.e., upon reaching the join or end of the process); (2) if the description of new paths should repeat or not information previously described by previous paths. Based on the identified consideration, each of the possible ways to describe the different paths generated by a split were identified.

As in question 4, *cases* were used to represent the different ways of describing the different paths generated by splits. Each *case* will be presented containing a specific notation, a description, and an example of process description.

For the notation, cases were defined as $Case_{5-i}$ where 5 represents the question number, and i represents an index associated with a given case (index starts with the value of 1 and will be incremented to represent each possible new case). In order to complement the description, each case will present an example of process description highlighting the paths covered by the description. The process descriptions created for each case are intended to describe the BPMN 2.0 process model with generic activities presented in Figure 3.3. This process model was created because it is an easy example to understand the different cases related to this question.

Figure 3.3: Question 5: Process model example.



Source: The authors

- $\mathbf{Case_{5-1}}$: Displays each possible path from the start of the split to the end of the process.
  - Example: When the process starts, one of the following paths is executed. *(Path 1)* If condition A is true, then activity 1 is performed. Then, if condition C is true, activity 4 is performed and the process ends. *(Path 2)* On the other hand, if condition A is true, then activity 1 is performed. Then, if condition D is true, then activity 3 is performed. Finally, activity 4 is performed and the process ends. *(Path 3)* On the other hand, if condition B is true, activity 2 is performed. Finally, activity 4 is performed and the process ends.
- $\mathbf{Case_{5-2}}$: Displays each possible path from the start of the split to the respective join.
  - Example: When the process starts, one of the following paths is executed. *(Path 1)* If condition A is true, then activity 1 is performed. Then, if condition C is true, the path ends. *(Path 2)* On the other hand, if condition A is true, then activity 1 is performed. Then, if condition D is true, then activity 3 is

performed. *(Path 3)* On the other hand, if condition B is true, activity 2 is performed. After performing one of the paths, activity 4 is performed and the process ends.

- $\mathbf{Case_{5-3}}$: Displays a possible path from the beginning of the split to the end of the process. It then presents the other paths taking into account what has already been presented.
  - Example: When the process starts, one of the following paths is executed. *(Path 1)* If condition A is true, then activity 1 is performed. Then, if condition C is true, activity 4 is performed and the process terminates. *(Path 2)* On the other hand, after activity 1, condition D may occur. Then activity 3 is performed. Finally, activity 4 is performed and the process ends. *(Path 3)* On the other hand, if condition B is true, activity 2 is performed. Finally, activity 4 is performed and the process ends.
- $\mathbf{Case_{5-4}}$: Displays a possible path from the beginning of the split to the respective join. It then presents the other paths taking into account what has already been presented.
  - Example: When the process starts, one of the following paths is executed. *(Path 1)* If condition A is true, then Activity 1 is performed. Then, if condition C is true, the path ends. *(Path 2)* On the other hand, after activity 1, condition D may occur. In this case, activity 3 is performed. *(Path 3)* On the other hand, if condition B is true, activity 2 is performed. After performing one of the paths, activity 4 is completed and the process ends.

In cases where a path is described and then presents the other paths taking into account what has already been described (i.e., case $\mathbf{Case_{5-3}}$ and $\mathbf{Case_{5-4}}$), they can prevent sentences describing previously presented paths from being repeated. In this way, this also contributes to a lesser process description, since an element already presented as a sentence will not be described again in another sentence. However, such cases may make it difficult to understand the process model as the reader can get lost in the fragmented paths of the text.

3.3 Analysis for Sentence Design

In order to determine the design of the sentences, an analysis of process descriptions in the literature was carried out to find the most recurrent sentence templates and to identify those with ambiguity issues. To the best of our knowledge, this work presents itself as the first effort to create a method to identify and classify sentence templates in business process descriptions. This section presents this method, as well as an analysis of the sentence templates identified in relation to ambiguity issues. A total of 101 sentence templates was found and divided into 29 categories. Each category is composed by sentence templates that can be replaced in a process description and represent the same information. In addition, six types of ambiguities were identified and, when compared with the sentence templates found, enabled us to define 13 templates related to ambiguity issues.

The following subsections present the procedures followed to: (3.3.1) prepare the sentences for the analysis, (3.3.2) identify and classify the sentence templates, and (3.3.3) address the ambiguity issues.

3.3.1 Preparation of Sentences

In this first stage, the sentences are prepared for the identification and classification of sentence templates. For this, the sentences of a process description are modified to become more generic. This modification is necessary because a business process description may contain snippets of text that are directly related to the process context. As an example, the sentences “The manager must sell the product” and “The salesman must sell the product” are identical, except by who carries out the activity of selling the product. This difference can hinder the identification and classification of sentence templates, since these sentences can be considered as different sentence templates. In this sense, a term capable of representing both “manager” and “salesman” could be used in order to make these two sentences equal and, consequently, to define both as the same sentence template. Thereby, the business process descriptions were previously analyzed and four different placeholders were created with the goal to replace in the sentences the snippets related to the context by more generic information. The created placeholders are: *role*, *condition*, *number*, and *object*.

The placeholder *role* is associated with the role responsible for performing a particular action. In relation to the business process model, a role could be considered as a participant. As an example, in the sentence “The process finishes after the technician completes the repair form.”, once the technician is the one performing the action, the word “technician” can be replaced by the placeholder *role*. Therefore, the sentence after the modification would be written as “The process finishes after the *role* completes the repair form.”.

The placeholder *condition* aims to define some condition that needs to be satisfied for a given flow to occur. Normally, the condition appears in a business process model as a label that tracks the output sequence flow of an exclusive or inclusive gateway. For instance, in the sentence “In case it is a software problem, the technician must format the computer.” it is possible to observe that to be done the activity of formatting the computer must exist before the condition “it is a software problem”. Therefore, this condition will be replaced in the text by the placeholder *condition*. Moreover, the technician can also be replaced in this sentence by the placeholder *role*.

The placeholder *number* is used to represents a certain amount of process elements or paths in a process model. As an example, in the sentence “After all five activities are completed, the process ends.”, the amount “five” can be replaced by the placeholder *number*.

Finally, the placeholder *object* can represent the business object to which the sentence refers. For instance, in the sentence “The car can be sold by the manager or the seller”, the business object “car” can be replaced by the placeholder *object*. In addition, the placeholders *role 1* and *role 2* could be created to represent the manager and seller respectively. After the preparation stage, the modified sentences containing the created placeholders will be used for the identification and classification of sentence templates.

3.3.2 Identification and Classification of Sentence Templates

In the context of this work, a sentence template was considered as each pattern presented in a sentence that is able to describe one or more process elements. These process elements appear in the template as placeholders to be replaced. For the scope of this work, a reduced set of notational elements is taken into account to find sentence templates, being: activity (A_c), AND-split (G_{+s}), AND-join (G_{+j}), XOR-split (G_{Xs}), XOR-join (G_{Xj}), OR-split (G_{Os}), OR-join (G_{Oj}), start event (E_s), intermediate event (E_i), end event (E_e). In addition, “empty” is used to define paths without elements (e.g., Figure 2.3, s_4).

Since a sentence template has been defined as a pattern presented in a sentence capable of describing one or more process elements, in order to identify the sentence template it is necessary to identify the process elements in the text. Although there are works that contribute to the automated identification of process elements in texts, to the best of our knowledge, there is no approach capable of extracting the process elements in a textual description with complete precision (EPURE et al., 2015; FERREIRA et al., 2017; FRIEDRICH; MENDLING; PUHLMANN, 2011). In addition, automated identification approaches can draw incorrect conclusions about a process by making assumptions about texts that allow for multiple interpretations (AA; LEOPOLD; REIJERS, 2016). Therefore, an automated analysis of sentence templates could be compromised by the selected approach of extracting process elements, so the identification of the sentence templates in the present work was carried out manually.

The identification and classification of sentence templates were carried out in parallel. To perform the classification of sentence templates, we define each sentence template as composed of three elements. These elements were defined because they describe the process characteristics present in a sentence template. These elements are:

- **Target**: the set of process elements described by the sentence template. A target must appear in the sentence, even if implicitly (i.e., without a placeholder to fill with the process element).

- **Relationship**: how the process elements in the sentence are associated to each other: none (R_N), composed by 0 or 1 process element; sequential (R_S), one element follows the other; exclusive (R_X), elements follow different paths that exit from the XOR-split gateway; inclusive (R_O), elements follow different paths that exit from the same OR-split gateway; and parallel (R_+), elements follow different paths that exit from the same AND-split gateway.

- **Source**: the process element that occurs immediately before the analyzed sentence. As in the BPMN 2.0 specification, the source can be understood as the element prior to the currently described element connected by a sequence flow. A source may or may not be evidenced in the sentence.

As an example, Figure 3.4 presents a process description adapted from the Figure 2.4. In this example, five sentence templates were identified, two of which have *target*

Figure 3.4: Example of process description: Computer repair (adapted from Figure 2.4).

(s_{a1}) The repair department of the company X performs repairs of computers and printers. (s_{a2}) Once a computer with problems arrives, the technician must perform an evaluation. (s_{a3}) In case it is a software problem, the technician must format the computer. (s_{a4}) For the case that it is a hardware problem, the technician must replace the part and fill out the part replacement form. (s_{a5}) This form must contain the part identification code and the technician's signature. (s_{a6}) On the other hand, if no problem is found, no modification should be made to the computer. (s_{a7}) The process finishes after the technician completes the repair form.

Source: The authors

with the sequential *relationship* (s_{a4}, s_{a7}) and three have *target* with the none *relationship* (s_{a2}, s_{a3}, s_{a6}). In the sentence s_{a2}, it is possible to define the sentence template "Once E_s, the *role* must A_c", where E_s is the *source* evidenced in the sentence template, A_c is a placeholder for an activity described in the *target* and *role* refers to some participant in the process that performs the activity A_c. The sentences s_{a3}, s_{a4} and s_{a6} have as *source* a XOR-split gateway not evidenced in the sentence template. In addition, the sentence s_{a7} has as its *source* a XOR-join gateway and as *target* an activity (A_c) and an end event (represented implicitly by "The process finishes after"). Not all sentences in a text are necessarily mapped to a sentence template, since process descriptions can be composed by other information, such as statements that contextualize the process (s_{a1}) and statements that detail an activity or business rule (s_{a5}).

In order to identify the sentence templates, each process description was inserted into a spreadsheet, as illustrated in Table 3.2. In the spreadsheet, each line represents one sentence and the columns represent the following attributes: sentence, sentence template ID (i.e., the ID of the sentence template that can be a number or "none") and sentence template.

After all the sentence templates were identified, they were grouped into categories based on *source*, *target* and *relationship*. As a result, each category is composed by sentence templates that can be replaced in a process description and represent the same information. As an example, the sentence s_{a3} is defined as a sequential *relationship* between a XOR-split (*source*) and an activity (*target*). This sentence can be rewritten by another sentence template that have the same properties, therefore the same category, such as: "Once *condition*, the *role* needs to A_c".

The analysis of sentence templates was done in two different manners, namely

Table 3.2: Example of identification of sentence templates.

Process Description: Computer repair		
Sentence	Sentence Template ID	Sentence Template
The repair department of the company X performs repairs of computers and printers.	None	
Once a computer with problems arrives, the technician must perform an evaluation.	3	Once E_s, the *role* must A_c.
In case it is a software problem, the technician must format the computer.	10	In case *condition*, the *role* must A_c.
For the case that it is a hardware problem, the technician must replace the part and fill out the part replacement form.	25	For the case *condition*, the *role* must A_c and A_c.
This form must contain the part identification code and the technician's signature.	None	
On the other hand, if no problem is found, no modification should be made to the computer.	1	On the other hand, if *condition*, $empty$.
The process finishes after the technician completes the repair form.	8	The process finishes after the *role* A_c.

Source: The authors

atomic level analysis and *group level analysis*. At the *atomic level analysis*, it is considered that if two sentences have the same text, but represent different process elements in the *source*, they are defined as two distinct sentence templates. For example, sentences "When a computer with problems arrives, the technician must perform an evaluation." and "When performing a repair, the technician must perform an evaluation." are defined as different sentence templates because they have different process elements in the source, being respectively: "When E_s, A_c." and "When A_c, A_c.". On the other hand, at the *grouped level analysis* it is considered that different possible process elements can be translated as the same sentence template. In this case, the two sentence templates described above can be viewed as a single sentence template (i.e., "When (A_c or E_s), A_c."), capable to have as *source* either an activity or a start event.

To facilitate the categorization of sentence templates, we create a notation based on the previously defined criteria. $St_i = R_s(source, target)$ can be interpreted as: there is a sentence template St_i that starts from a $source$, can describe a $target$ and is associated through a sequential relationship R_s. In the case of *atomic level analysis*, a $source$ is a process element. On the other hand, in the case of *group level analysis* a $source$ is a set of possible process elements (e.g., $A_c|G_{Xs}|G_{+s}$). A target can be described as $target =$

$R_x(component_1, ..., component_n)$, i.e., a target is a set of components that relate to each other through a relationship R_x. Finally, a component can be a process element, *empty*, or another target. Thus, two different sentence templates belong to the same category if they share the same notation, which means to start from the same $source$ and reach the same $target$.

3.3.3 Ambiguity in Sentence Templates

After the identification and classification of the sentence templates, they were analyzed in relation to ambiguity issues. As mentioned, in the context of this work a sentence template was considered ambiguous when it allows multiple interpretations of the process. To identify common ambiguity issues in process descriptions, two approaches were carried out: *analysis of the literature* and *analysis of the sentence templates*.

As for the *analysis of the literature*, works related to ambiguity in process descriptions were investigated. Although some works related to this subject were found, only a few of them (AA; LEOPOLD; REIJERS, 2015; AA; LEOPOLD; REIJERS, 2016; AA; LEOPOLD; REIJERS, 2018; AKBAR; BAJWA; MALIK, 2013) presented cases of ambiguity. This analysis of the literature made it possible to find eight ambiguity problems that were categorized into five different types of ambiguity.

In terms of the *analysis of the sentence templates*, two independent tasks were conducted. In the first part, an analysis of each sentence template was carried out individually in order to identify ambiguity issues. In the second part, an analysis was carried out involving the combination of sentence templates. For the latter case, sentence templates that share the same description, but do not have the same classification (i.e., *source*, *target* or *relationship*) were considered candidates for ambiguity issues.

Table 3.3 presents the six different types of ambiguities that were identified in this work, with their respective identifiers ($Ambi_{ID}$), descriptions, examples, and source.

3.3.4 Analysis of Sentence Templates and Ambiguity Issues

After analyzing the process descriptions in an *atomic level*, it was possible to obtain a set of 101 sentence templates for 29 categories. Of these, 13 sentence templates were classified as having one of the six ambiguity issues. Tables 3.4 to 3.6 show the sen-

Table 3.3: Identified ambiguity issues.

Ambi_{ID}	**Description**	**Example**	**Source**
$Ambi_1$	The term "and" can have different meanings, such as: sequence, dependence, parallelism, contrast.	*The employee must update the document* ***and*** *prepare the product for shipping.*	- Akbar, Bajwa and Malik (2013) - Aa, Leopold and Reijers (2016) - Aa, Leopold and Reijers (2018)
$Ambi_2$	The terms "or" and "sometimes" may raise doubts whether it includes or is mutually exclusive to the different alternatives.	*(1) The document is accepted* ***or*** *denied.* *(2) The bicycle can be mounted* ***or*** *painted.*	- Akbar, Bajwa and Malik (2013) - this work
$Ambi_3$	The term "latter" usually does not make clear to what previous activities it refers.	*In parallel to the* ***latter*** *steps...*	- Aa, Leopold and Reijers (2016) - Aa, Leopold and Reijers (2018)
$Ambi_4$	The terms "meanwhile", "concurrently", "meantime", "in the meantime" and "at the same time" make it difficult to specify which sets of activities they refer to.	***In the meantime****, the sales department must prepare the receipt.*	- Aa, Leopold and Reijers (2018) - this work
$Ambi_5$	Repetitions usually not clear what activities should be performed again.	*The previous steps must be* ***repeated****.*	- Aa, Leopold and Reijers (2015) - Aa, Leopold and Reijers (2016) - Aa, Leopold and Reijers (2018)
$Ambi_6$	The term "while" can mean simultaneity or concession.	***While*** *it is true that the company has the money, they can't build the houses.*	- this work

Source: The authors

tence templates for each one of the 29 categories (C_{ID}), with their respective category notation. Each sentence template has a specific identifier presented in the "St_{ID}" column. In addition, the number of times each sentence template appeared in the process descriptions analyzed is presented in the "N" column. Moreover, the ambiguity issues identified for each sentence template, when identified, is presented in the "$Ambi_{ID}$" column, based on the elements in Table 3.3.

Considering the identified sentence templates, the most recurrent is "If *condition*, A_c." (i.e., St_{71}), from category C_{18}, which appeared 81 times. It is possible to observe that this sentence template is fairly recurrent in business process descriptions because the two

Table 3.4: Atomic sentence templates by category 1.

C_{ID}	Notation	St_{ID}	Sentence Template	N	$Ambi_{ID}$
C_1	$R_S(A_c, R_N(A_c))$	St_1	Once A_c, A_c.	15	
		St_2	Then A_c.	13	
		St_3	When A_c, A_c.	11	
		St_4	After A_c, A_c.	10	
		St_5	Next A_c.	5	
		St_6	Afterwards, A_c.	4	
		St_7	As soon as A_c, A_c.	3	
		St_8	Subsequently A_c.	3	
		St_9	The *role* then A_c.	3	
		St_{10}	Upon A_c, A_c.	3	
		St_{11}	After that A_c.	2	
		St_{12}	Likewise A_c.	2	
		St_{13}	A_c, after which A_c.	1	
		St_{14}	Immediately after A_c, A_c.	1	
		St_{15}	In addition to A_c, A_c.	1	
		St_{16}	In the following A_c.	1	
		St_{17}	Moreover, A_c.	1	
		St_{18}	Thereafter A_c.	1	
		St_{19}	Therefore A_c.	1	
C_2	$R_S(A_c, R_N(E_e))$	St_{20}	After A_c, the process ends.	1	
		St_{21}	After all *number* activities are completed the process ends.	1	
		St_{22}	For *role* the process ends then.	1	
		St_{23}	The process ends here.	1	
		St_{24}	The process finishes only once A_c.	1	
C_3	$R_S(A_c, R_N(G_{Xs}))$	St_{25}	After A_c, the *object* may lead to *number* possible outcomes: *condition* or *condition*.	2	$Ambi_2$
		St_{26}	After A_c, it is checked whether *condition*.	1	
		St_{27}	After A_c, the *role* can either *condition*, *condition* or *condition*.	1	
		St_{28}	After A_c, the *role* investigates whether *condition* or *condition*.	1	
		St_{29}	After A_c, the *role* may either *condition* or *condition*.	1	
		St_{30}	One of the *number* alternative process paths may be taken.	1	
		St_{31}	The *role* can either *condition* or *condition*.	1	
		St_{32}	The *role* can then *condition* or *condition*.	1	$Ambi_2$
		St_{33}	This procedure is repeated for each *condition*.	1	$Ambi_5$
		St_{34}	When A_c, it is first checked whether *condition*.	1	

Source: The authors

Table 3.5: Atomic sentence templates by category 2.

$\mathbf{C}_{\mathbf{ID}}$	**Notation**	$\mathbf{St}_{\mathbf{ID}}$	**Sentence Template**	**N**	$\mathbf{Ambi}_{\mathbf{ID}}$
C_4	$R_S(A_c, R_N(G_{Xj}))$	St_{35}	Then the process continues normally.	1	
C_5	$R_S(A_c, R_S(A_c, A_c))$	St_{36}	A_c and A_c.	15	$Ambi_1$
		St_{37}	Also A_c and A_c.	1	$Ambi_1$
C_6	$R_S(A_c, R_S(A_c, A_c, A_c))$	St_{38}	A_c and A_c and A_c.	1	$Ambi_1$
		St_{39}	First A_c, then A_c, and finally A_c	1	
C_7	$R_S(A_c, R_S(A_c, E_e))$	St_{40}	A_c, which ends the process.	2	
		St_{41}	Finally, A_c.	2	
		St_{42}	The process completes with A_c.	2	
		St_{43}	A_c, then the process ends.	1	
		St_{44}	After A_c, this process path ends.	1	
		St_{45}	Afterwards, A_c and finishes the process instance.	1	
		St_{46}	The process finishes when A_c.	1	
C_8	$R_S(A_c, R_S(A_c, G_{Xs}))$	St_{47}	The *role* A_c and may decide to either A_c or A_c.	1	$Ambi_1$
C_9	$R_S(A_c, R_+(A_c, A_c))$	St_{48}	While A_c, A_c.	2	$Ambi_6$
		St_{49}	Next, A_c while A_c.	1	$Ambi_6$
		St_{50}	Once A_c, A_c and meantime A_c.	1	
C_{10}	$R_S(A_c, R_X(A_c, A_c))$	St_{51}	After A_c, the *role* either A_c or A_c.	1	
		St_{52}	When A_c, the *role* may either A_c or A_c.	1	
C_{11}	$R_S(A_c, R_X(A_c, A_c, A_c))$	St_{53}	Sometimes A_c, sometimes A_c and sometimes A_c.	1	$Ambi_2$
C_{12}	$R_S(A_c, R_O(A_c, A_c))$	St_{54}	*object* may be A_c from either *role 1* or *role 2* or from both.	1	
		St_{55}	The *role* may either A_c or also A_c.	1	
C_{13}	$R_S(E_i, R_N(E_e))$	St_{56}	After *role* E_i, the process flow ends.	1	
C_{14}	$R_S(E_s, R_N(A_c))$	St_{57}	The process starts with A_c.	7	
		St_{58}	The process starts when A_c.	6	
		St_{59}	First, A_c.	3	
		St_{60}	The process starts by A_c.	2	
		St_{61}	When E_s, A_c.	2	
		St_{62}	Whenever E_s, A_c.	2	
		St_{63}	After the process starts, A_c.	1	
		St_{64}	The process is triggered by A_c.	1	
		St_{65}	The process starts once A_c.	1	
C_{15}	$R_S(E_s, R_S(A_c, A_c, A_c))$	St_{66}	A_c and A_c and A_c.	1	$Ambi_1$
C_{16}	$R_S(G_{+s}, R_N(A_c))$	St_{67}	In the meantime, A_c.	2	$Ambi_4$
		St_{68}	At the same time, A_c.	1	$Ambi_4$
C_{17}	$R_S(G_{+j}, R_N(A_c))$	St_{69}	Afterwards, A_c.	2	
		St_{70}	After each of these activities, A_c.	1	

Source: The authors

Table 3.6: Atomic sentence templates by category 3.

$\mathbf{C_{ID}}$	**Notation**	$\mathbf{St_{ID}}$	**Sentence Template**	**N**	$\mathbf{Ambi_{ID}}$
C_{18}	$R_S(G_{Xs}, R_N(A_c))$	St_{71}	If *condition*, A_c.	81	
		St_{72}	Otherwise, A_c.	10	
		St_{73}	In this case A_c.	7	
		St_{74}	In case *condition*, A_c.	3	
		St_{75}	For the case *condition*, A_c.	2	
		St_{76}	*condition*, in which case A_c.	1	
		St_{77}	*condition*, otherwise A_c.	1	
		St_{78}	However, if *condition*, A_c.	1	
		St_{79}	In that case, A_c.	1	
		St_{80}	In the latter case, A_c.	1	
		St_{81}	On the other hand, if *condition* A_c.	1	
		St_{82}	Sometimes *condition*, then A_c.	1	
C_{19}	$R_S(G_{Xs}, R_N(E_e))$	St_{83}	If *condition* the process will end.	1	
		St_{84}	In the former case, the process instance is finished.	1	
C_{20}	$R_S(G_{Xs}, R_N(G_{+s}))$	St_{85}	This action consists of *number* activities, which are executed in an arbitrary order.	1	
C_{21}	$R_S(G_{Xs}, R_N(G_{Xs}))$	St_{86}	If *condition*, this results in either *condition* or *condition*.	1	
C_{22}	$R_S(G_{Xs}, R_S(A_c, A_c, A_c))$	St_{87}	If *condition*, *role* may need to first A_c, then A_c and finally A_c.	1	
C_{23}	$R_S(G_{Xs}, R_+(A_c, A_c))$	St_{88}	Once *condition*, A_c and meantime A_c.	3	
C_{24}	$R_S(G_{Xs}, R_X(A_c, A_c))$	St_{89}	If *condition*, A_c, otherwise A_c.	4	
		St_{90}	In case *condition*, A_c, otherwise A_c.	1	
		St_{91}	*role* either A_c or A_c.	1	
C_{25}	$R_S(G_{Xs}, R_X(A_c, E_e))$	St_{92}	If *condition* A_c, otherwise the process is finished.	1	
C_{26}	$R_S(G_{Xs}, R_X(A_c, empty))$	St_{93}	If *condition*, A_c, except if *condition*.	1	
		St_{94}	In case *condition*, A_c otherwise the process continues.	1	
C_{27}	$R_S(G_{Xj}, R_N(A_c))$	St_{95}	In any case, A_c.	4	
		St_{96}	In either/any case, A_c.	1	
		St_{97}	Once one of these *number* activities is performed, A_c.	1	
		St_{98}	The process then continues with A_c.	1	
C_{28}	$R_S(G_{Xj}, R_S(A_c, E_e))$	St_{99}	Afterwards, A_c and the process completes.	1	
		St_{100}	Finally, A_c.	1	
C_{29}	$R_S(G_{Xj}, R_+(A_c, A_c))$	St_{101}	Then, two current activities are triggered, A_c and A_c.	1	

Source: The authors

sentence templates that appear the most after this sentence template were identified only 15 times (i.e., St_1 and St_{36}). Moreover, the category that presented the largest diversity of sentence templates is C_1, with 19 distinct sentence templates, followed by C_{18} (with 12) and C_3 (with 10).

In terms of ambiguity, the type that appeared most in the sentence templates is related to the term “and” ($Ambi_1$), having occurred five times. Moreover, $Ambi_2$ appeared three times, followed by both $Ambi_4$ and $Ambi_6$ (2 times), and $Ambi_5$ (1 time). In addition, in the identified sentence templates no case was found related to the ambiguity $Ambi_3$.

Furthermore, it is possible to notice that not all relationships between process elements are explored in Tables 3.4 to 3.6. This occurs because some relationships that occur in the model are not explicitly transformed into sentences. Also, there are some relationships that appear less frequently than others in the texts considered.

In the *group level analysis*, the atomic level sentence templates that share the same target but presents different process elements as sources were grouped. From the data collected in the *atomic level analysis*, it was possible to identify 8 sentence templates that were transformed into four grouped sentence templates. Table 3.7 presents the grouped sentence templates. In this table are presented the new notations able to represent the grouped sentence templates, as well as the new sentence templates. In addition, the identifier of the atomic sentence templates used in each grouped sentence template are presented in the “St_{ID}” column. Finally, as in *atomic level analysis*, the number of times each grouped sentence template appeared in the process descriptions analyzed is presented in the “N” column.

Table 3.7: Grouped sentence templates.

Notation	**Sentence Template**	**St$_{\mathbf{ID}}$**	**N**
$R_s(A_c \vert E_s, Rn(A_c))$	When $A_c \vert E_s$, A_c.	St_3, St_{61}	13
$R_s(A_c \vert G_{+j}, Rn(A_c))$	Afterwards, A_c.	St_6, St_{69}	6
$R_s(A_c \vert G_{Xj}, R_s(A_c, E_e))$	Finally, A_c.	St_{41}, St_{100}	3
$R_s(A_c \vert E_s, R_s(A_c, A_c, A_c))$	A_c and A_c and A_c.	St_{38}, St_{66}	2

Source: The authors

3.4 Text Design

To perform the text design, the analyses performed previously will be used to define how business process-oriented texts will be constructed. For this, design decisions will be presented taking into account the five questions raised during the analysis for structuring the text and the sentences selected during the analysis for sentence design. This section will also present other design decisions and their respective justifications.

The following subsections present the steps followed to: (3.4.1) define the term to represent the different paths of the process, (3.4.2) define the text structure according to the analysis for structuring the text performed, and (3.4.3) define the sentences that will compose the text according to the analysis for sentence design performed.

3.4.1 Term to Represent Processes Paths

To represent different paths generated by a split, different terms can be used, such as: “procedures”, “paths”, “branches”, “set of elements”, “activities”. In order to normalize the different sentences that will compose the process description, we chose to use the term “procedure” whenever referring to these different paths. In this regard, for the construction of the process description, a sentence template such as “One of the *number* alternative process paths may be taken.” (i.e., St_{30}) was transformed into “One of the *number* alternative *procedures* may be taken.”.

The term “procedure” was chosen because the terms “paths” and “branches” present the idea of flows in a process model. As the purpose is to describe a process even by those who do not have understanding in process modeling, the term “procedure” seems more appropriate. In addition, the terms “set of elements” and “activities” may give an impression of arbitrariness in the actions to be carried out while “procedure” has more the idea of not only what must be done but how must be done (i.e., how to proceed).

3.4.2 Text Structuring Design

In this section the structure of the text is defined according to the analysis for structuring the text carried out in Section 3.2. In relation to how the text will be described in relation to the process (Question 1), we chose to describe the text sequentially (i.e., starting from the initiating events and going to the end events) because this was the most recurrent way of describing processes in business process descriptions analyzed. Concerning how the text is organized in terms of paragraphs (Question 2), it was chosen, as well as Ferrari et al. (2017), to describe paragraphs with a maximum of 5 sentences. However, recognizing that different factors may contribute to the decision of paragraph size, the architecture presented in Chapter 4 of this work seeks to provide ways to customize the size of paragraphs. Regarding the voice used in the text (Question 3), the text will always be presented in the active voice because this form makes explicit the actor who performs a certain action. If it is not possible to define the actor of a particular action, a generic actor (e.g., “Resource 1”) should be created.

Regarding how the text will describe splits and joins (Question 4), the use of sentences to explicitly present splits and joins make text more verbose. However, it facilitates in identifying where splits and joins occur in the process. Thus, we chose to maintain sentences that describe splits and joins. Regarding splits, sentences will explicitly describe splits (i.e., $\mathbf{Case_{4-s1}}$). In relation to joins, there are two cases able to make explicit the joins in the text. Among these cases, markers will be used to make explicit in the text where there is a join (i.e., $\mathbf{Case_{4-j3}}$).

In order to describe the different paths of a split (Question 5), two cases were chosen. First, it was chosen to describe each possible path from the beginning of the split to the respective join (i.e., $\mathbf{Case_{5-2}}$). However, considering that a process can contain several paths and that this can lead to very long descriptions where many sentences are repeated for the different paths, the case that describes from split to join without repeating what has already been described can be used (i.e., $\mathbf{Case_{5-4}}$).

3.4.3 Sentence Design

In relation to the sentence analysis, the sentence templates were chosen taking into consideration three criteria, being applied in the order in which they appear:

1. **Sentence templates without ambiguity issues:** Sentence templates that are not ambiguous considering the cases of ambiguity previously pointed.

2. **Non-generic sentence templates:** Sentence templates that are not dependent on the source (e.g., Once A_c, the *role* must A_c).

3. **Most recurring sentence templates:** Sentence templates that appear with recurrence in business process descriptions.

Table 3.8 presents the sentence templates chosen to compose the process description. This table contains the element being considered, the type of the element, which categories of the sentence templates match the respective element and type, the chosen sentence template, and the corresponding sentence template ID (i.e., St_{ID}). In addition to the sentence templates chosen from the analysis, some sentence templates were adapted (i.e., use of the term "(adapt)" in St_{ID}) and others were created (i.e., use of the term "Created" in St_{ID}).

In relation to *start* element, were selected sentence templates to describe the start of a process. This type of sentence can be find in the categories C_{14} and C_{15} present in Table 3.5. The sentence templates were selected according to two types: *sequence* and *split*. Regarding *sequence*, a sentence template was selected that demonstrates a sequencing between the beginning of the process and an activity. For this, the sentence template St_{57} of the Table 3.5 was selected. This sentence template could be complemented with an activity such as "The process starts with *a client making a request.*". On the other hand, in relation to the type *split* a sentence template was selected that demonstrates the sequencing between the beginning of the process and a split gateway. For this, the sentence template St_{61} was adapted and used. This sentence template could be complemented with a sentence that represents a split gateway, for example: "When the process starts, *one of the number alternative procedures is executed.*".

Regarding *sequence* element, sentence templates that represent the sequence of actions were selected. For this, was selected templates of two types: *atomic* and *aggregation*. The type *atomic* describes an activity that proceeds another. Elements of this type are represented in Table 3.4 as in category C_1. Since *atomic* is quite recurrent in process descriptions, three sentence templates of this type have been selected. The three sentence templates selected are those that do not present ambiguity issues, are non-generic, and are most recurring. In addition, the sentence templates "Then, A_c." (i.e., St_2) and "Afterwards, A_c." (i.e., St_6) were not used because these terms will be present in other sentence templates. On the other hand, the type *aggregation* seeks to represent sentence templates that contain more than one activity being described in sequence. The most recurrent sentence template in this group is St_{36} (i.e., "A_c and A_c."). However, this sentence template presents an ambiguity issue. Thus, based on the sentence templates St_{39} and St_{87}, the sentence template "A_c and then A_c." was created that clarifies the idea of sequencing between activities.

In relation to *end* element, were selected sentence templates to describe the end of a process. This type of sentence can be found in several categories present on Tables 3.4 to 3.6. The sentence templates were selected according to two types: *sequence* and

Table 3.8: Sentence templates chosen to compose the process description.

Element	**Type**	**Categories**	**Sentence Template**	**St$_{\text{ID}}$**
Start	Sequence	C_{14}, C_{15}	The process starts with ...	St_{57}
	Split	C_{14}, C_{15}	When the process starts, ...	St_{61} (adapt)
Sequence	Atomic 1	C_1	Next, ...	St_5
	Atomic 2	C_1	Subsequently, ...	St_8
	Atomic 3	C_1	After that, ...	St_{11}
	Aggregation	C_5, C_5, C_{15}, C_{22}	... A_c and then A_c.	St_{39}, St_{87} (adapt)
End	Sequence	C_2, C_7, C_{13}, C_{19}, C_{25}, C_{28}	Finally, the process ends.	St_{20}, St_{41} and St_{100} (adapt)
	Join	C_2, C_7, C_{13}, C_{19}, C_{25}, C_{28}	..., the process ends.	St_{20}
Exclusive Choice	Split	C_3	..., one of the *number* alternative procedures is executed.	St_{30} (adapt)
	Join	C_4, C_{27} - C_{29}	In any case, . . .	St_{95}
	Path	C_{18} - C_{26}	In the *ordinal* procedure, ...	Created
	Path with condition	C_{18} - C_{26}	If "*condition*", ...	St_{71}
Inclusive Choice	Split	None	..., *number* alternative procedures may be executed.	St_{30} (adapt)
	Join	None	Afterwards, . . .	St_{69} (adapt)
	Path	None	In the *ordinal* procedure, ...	Created
	Path with condition	None	If "*condition*", ...	St_{71} (adapt)
Parallelism	Split	C_{16}, C_{20}	..., *number* procedures are executed in an arbitrary order.	St_{85} (adapt)
	Join	C_{17}	After each case, ...	St_{95} (adapt)
	Path 1	C_{16}	In the meantime, ...	St_{67}
	Path 2	C_{16}	At the same time, ...	St_{68}

Source: The authors

join. Regarding the type *sequence*, a sentence template was selected that demonstrates a sequencing between an activity and the end of the process. In relation to the type *join*, a sentence template was selected that demonstrates the sequencing between a join gateway and the end of the process. In both cases, the sentence template evidenced in St_{20} "...the process ends." was used. Although it appears only once, similar sentence templates have appeared elsewhere (i.e., St_{21}, St_{22}, St_{23}, St_{40}, St_{43}, St_{44}, St_{56}, and St_{83}). The difference between the two types is in the preceding snippet. For the type *sequence* the sentence template presents in St_{41} and St_{100} "Finally, ..." was selected to precede thus forming the sentence "Finally, the process ends.". On the other hand, for the type *join* the sentence template that will precede must come from a join gateway, for example: "In any case, the process ends.".

The next three elements are related to gateways. To represent them, the following types will be used: *split*, *join*, *path*, and *path with condition*. For *split* type and *join* type, sentence templates will be used that represent, respectively, split gateways and join gateways. For the *path* type will be used sentence templates that describe the different paths generated by split gateways. In addition, a particular case of *path* named *path with condition* has been created. This particular case will be applied whenever all possible paths of a gateway have a related condition. In terms of BPMN 2.0 process model, it would be the equivalent of all sequence flows that are output from a split gateway having a label that describes a condition.

Regarding the exclusive choice, were selected sentence templates to describe the XOR gateway. The *split* sentence template was chosen because it is the first that meets the three criteria (more specifically, it is the first that is non-generic). In addition, the sentence template was adapted to describe *procedures* rather than *process paths*. Moreover, the *join* and *path with condition* sentence templates were chosen because they also met the defined criteria. Finally, to create sentences of type *path* a sentence template containing the term *ordinal* was created. This term should be modified to the ordinal of each procedure as described (i.e., first, second, third, etc.). Thus, an example of a sentence with type *path* would be "In the *first* procedure, the manager will request the report.".

For inclusive choice, were selected sentence templates to describe the OR gateway. In this case, no sentence template was found to fill the conceived categories. In the descriptions observed, few cases of sentence templates involving the OR gateway were found. The case found is present in the category C_{12}. However, this category represents sentences that have multiple activities being performed as inclusive and, in turn, do not fit into any of the expected cases. Thus, the sentence template St_{30} was adapted to be used by type *split* and the sentence template St_{69} was used by type *join*. In addition, just like the exclusive choice, was used the created sentence template to represent the type *path*. Finally, the sentence template for type *path with condition* was copied from the exclusive choice, since this sentence template can be used for both cases. Regarding parallelism, were selected sentence templates to describe the AND gateway. The *split* sentence was adapted to present the *procedures* instead of the *activities*. Regarding *join*, even though there is a particular category of sentences, the defined sentence was chosen to standardize with the sentence St_{95}. Since there are no conditions in AND gateways, two types of sentences have been defined to represent the multiple paths of a parallelism: "In the meantime, ..." (i.e., St_{67}) and "At the same time, ..." (i.e., St_{68}). As can be seen in Table 3.5, these two sentence templates are considered ambiguous because they do not make clear what is being executed in parallel. However, the use of explicit *split* and *join* sentences may help mitigate this issue.

3.5 Study Case: Computer Repair Process Description

As a demonstration of the design of business process descriptions as presented in this work, the process description described in Figure 3.4 could be rewritten as shown by Figure 3.5. This example was written manually from the recommendations proposed in Section 3.4.

Figure 3.5: Example of process description of Figure 3.4 rewritten.

(s_{a1}) The repair department of the company X performs repairs of computers and printers. (s_{a2}) The process starts with the technician performing an evaluation. (ns_1) Then, one of the 3 alternative procedures is executed.

(s_{a3}) If "it is a software problem", the technician must format the computer. (s_{a4}) If "it is a hardware problem", the technician must replace the part *and then* fill out the part replacement form. (s_{a5}) This form must contain the part identification code and the technician's signature. (s_{a6}) If "no problem is found", no modification should be made to the computer.

(ns_2) In any case, the technician completes the repair form. (ns_3) Finally, the process ends.

Source: The authors

The first thing to note is that the process description has been separated into three paragraphs. For this case, this division was chosen because it separates the elements that are outside the gateway of those that are inside the gateway. In addition, no paragraph was too long (i.e., with more than 5 sentences).

In this new process description, the sentence s_{a1} remained the same because it only presents context information. In addition, the sentence s_{a2} was modified by the sentence template "The process starts with A_c" (i.e., St_{57}). Then, to make the split gateway explicit, the sentence "Then, one of the 3 alternative procedures is executed." (i.e., ns_1) was created consisting of the sequencing sentence template "Then, ..." (i.e., St_2) and the sentence template for exclusive choice "..., one of the *number* alternative procedures is executed." (i.e., St_{30} (adapt)).

Moreover, the sentences s_{a3}, s_{a4} and s_{a6} were rewritten using the sentence template "If "*condition*", ..." (i.e., St_{71}). In addition, to solve the ambiguity issue related to "and" (i.e., $Ambi_1$) in the sentence s_{a4}, the sentence was modified to the sentence template "... A_c and then A_c". Furthermore, the sentence s_{a5} also has not been modified because this sentence only describes a business rule.

Finally, the sentence s_{a7} was separated into two new sentences: ns_2 and ns_3. The first sentence (i.e., ns_2) was created to make the join explicit and to represent the last activity in this process. On the other hand, the second sentence (i.e., ns_3) was created to represent the end of the process.

3.6 Final Considerations

In this chapter, two analyses were performed to understand how process descriptions are usually presented. Based on these analyses, it was defined how the process descriptions should be described taking into account the information collected in the previous analyses. Finally, an example of a process description created from the design decisions suggested in this chapter are presented.

Regarding the structuring of the text, it sought to understand how texts are usually structured to display the information of a business process. In order to answer this question, we have defined 5 questions about process descriptions. To answer these questions, analyses were performed on actual process descriptions and on different approaches in the literature.

Regarding analysis of sentence design, an empirical analysis of business process descriptions was carried out in order to discover the most recurrent sentences used to describe BPMN process models. In addition, an analysis was performed in order to find sentences with ambiguous meaning. The analysis consisted of three different steps. First, a set of 64 process descriptions was selected and their sentences were prepared for the identification of sentence templates. Then, an identification and a classification of sentence templates was performed in the prepared sentences. Finally, the sentence templates were marked as having or not ambiguity issues. Among all, 101 sentence templates were found and they were classified into 29 different categories. Of these, 13 sentence templates were considered as having ambiguity issues.

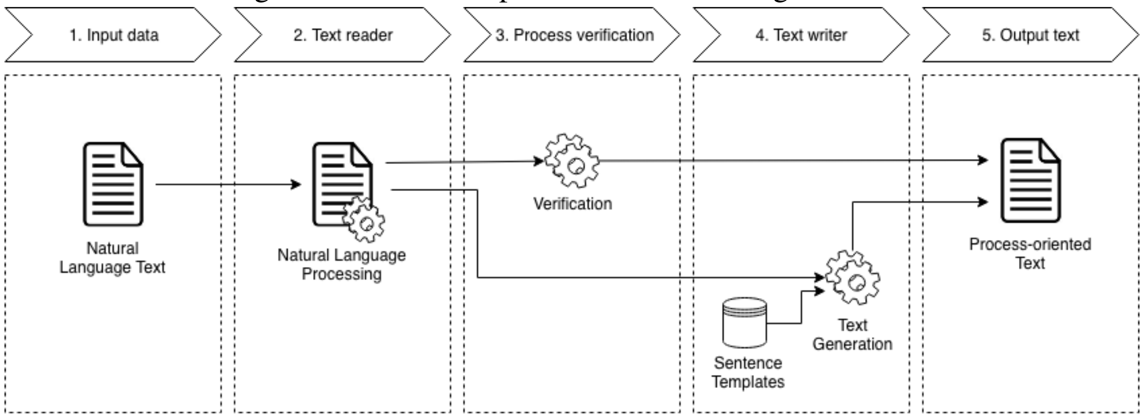
This chapter aims to contribute to the description of business processes in a way that is closer to a pattern and with less ambiguity issues. It can be useful for creating process descriptions more suitable for process analysts and domain experts. In addition, the results raised by this chapter can help approaches for *identification of process elements in natural language texts* and for *automated creation of business process descriptions*. For *identification of process elements in natural language texts*, the approaches can use the identified sentence templates as patterns to be sought in texts. In this case, sentence templates could be searched in the sentences of a process description. By finding a sentence corresponding to a sentence template, the process elements present in the sentence could be identified. For *automated creation of process descriptions*, the approaches can choose to use the most recurring sentence templates, or take advantage of the variety of sentence templates in each category to make the text more diversified.

In the next chapter will be presented the architecture developed for the generation of business process-oriented text.

4 ARCHITECTURE FOR BUSINESS PROCESS-ORIENTED TEXT GENERATION

In order to be able to generate a business process-oriented text, we construct an approach that consists of 5 stages: *input data*, *text reading*, *process verification*, *text writing* and *text output*. Firstly, a natural language text is given as input to the input data (i.e., *input data stage*). Then the approach reads the natural language text and produces an intermediate structure (i.e., *text reader stage*). Next, the intermediate structure is used to verify the process described by the text in relation to the BPMN 2.0 and in relation to soundness (i.e., *process verification stage*), and to generate the process description (i.e., *text writer stage*). Finally, the verification and the generated process description are combined for the generation of the business process-oriented text (i.e., *output text stage*). Figure 4.1 presents the approach with its respective stages.

Figure 4.1: Business process-oriented text generation.



Source: The authors

The approach described in this work proposes to generate business process-oriented text with the following set of BPMN 2.0 process elements as scope: task, subprocess, XOR gateway, AND gateway, OR gateway, start event, intermediate event, end event, sequence flow, lane, pool.

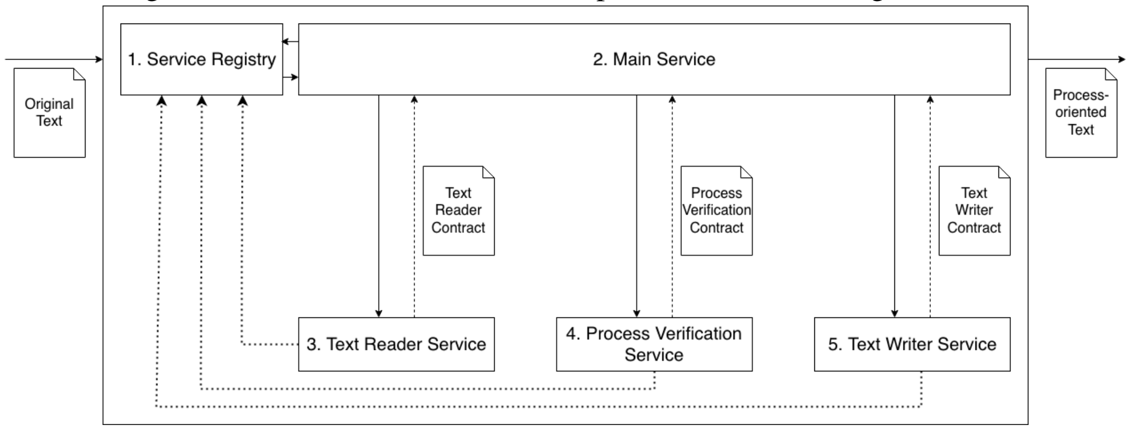
The remainder of this chapter is organized as follows: Section 4.1 presents an overview of the architecture built based on the described approach (see. Figure 4.1). In this section, are presented the services and the interaction between them for the generation of a business process-oriented text. Then, Sections 4.2 and 4.3 present the files used for defining the contracts. In this sense, Section 4.2 presents the data definition files and the Section 4.3 presents the service definition files. Furthermore, Sections 4.4 to 4.7 presents the services defined in this work. Finally, Section 4.8 presents the final considerations of this chapter.

4.1 Architecture Overview

The architecture developed in this work follows the principles of the Service Oriented Architecture (SOA) (ERL, 2008). This style of software architecture has been chosen because it allows to decompose the approach into services that have autonomy, are self-managed, and can communicate independently of the programming language, software versions or operational systems. Figure 4.2 presents the architecture overview composed of 5 services, being them: *Service Registry*, *Main Service*, *Text Reader Service*, *Process Verification Service*, and *Text Writer Service*. Each service is responsible for playing a role in generating the business process-oriented text:

- **Service Registry:** Service that maintains the location (URI) of the available services.
- **Main Service:** Service that acts as a service consumer making requests to the other services. Based an input (i.e., business process description or BPMN 2.0 process model), this service interacts with other services to produce the business process-oriented text.
- **Text Reader Service:** Service responsible for identifying the process elements and their respective relationships in a business process description. This service provides two functionalities:
    1. Generation of a intermediate structure containing the process elements and their relationships.
    2. Generation of metadata that can be used to mark and structure the process elements in the original text.
- **Process Verification Service:** Service responsible for performing the process verification. It verifies that the process is in accordance with the BPMN 2.0 and helps to verify the soundness of the process.
- **Text Writer Service:** Service responsible for generating the text according to the text design defined in Chapter 3.

Figure 4.2: Architecture for business process-oriented text generation.



Source: The authors

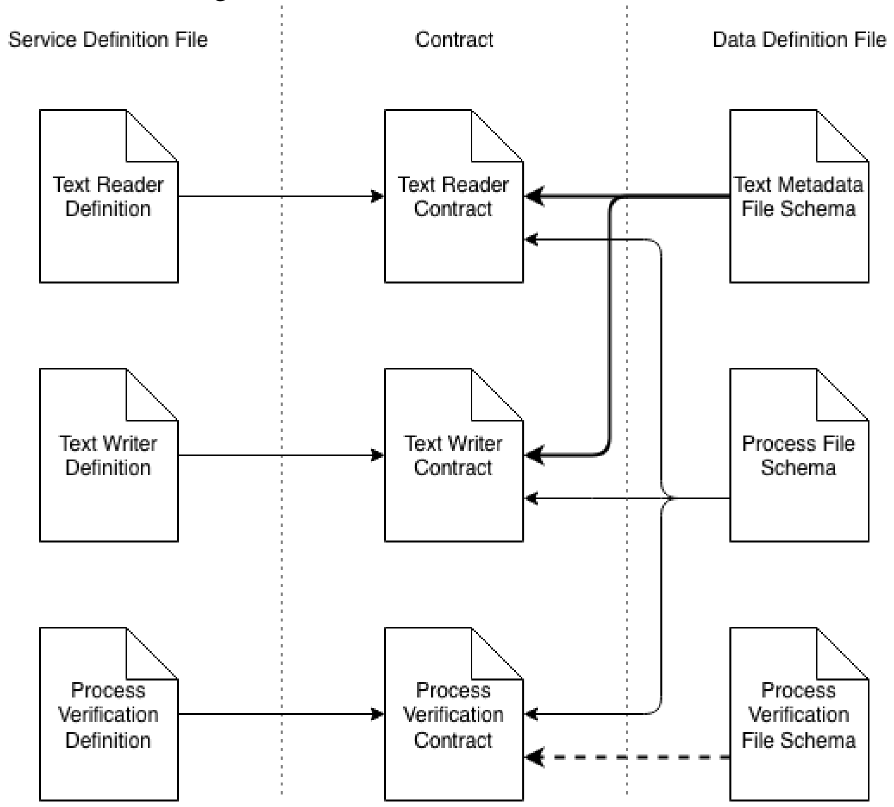
Some of the services presented in this work use or are adaptations of other approaches present in the literature. In this context, when a service makes use of an existing approach, a description of the approach, the reasons for its use, and the main modifications made to suit the purpose of the work will be presented.

In terms of implementation, this work followed the contract first approach (ERL, 2008). In this approach, it is first defined how the contracts will be and then it is defined how the services will communicate with those contracts. This approach allows services to be coupled to contracts (i.e., service-contract coupled) rather than the opposite. Therefore, the contracts are not created from the service logic and, hence, make the architecture independent of the information referring to the context of that service. Thus, the services were defined and prepared so that they could communicate according to the definitions of the contracts.

As can be observed in Figure 4.2, the services presented in the architecture communicate through three different contracts, being: *Text Reader Contract*, *Process Verification Contract*, and *Text Writer Contract*. These contracts define how to communicate with the *Text Reader Service*, *Process Verification Service*, and *Text Writer Service* respectively.

In the context of this work, each of the defined contracts consists of two different types of files: *data definition files* and *service definition files*. The *data definitions files* are documents that define the data that will be shared between the services. In total, three *data definitions files* were created, being: *Process File Schema*, *Text Metadata File Schema*, and *Process Verification File Schema*. On the other hand, the *service definition files* are documents that represent how services will communicate (i.e., possible requests and responses). In total, three *service definition files* were created, being: *Text Reader*

Figure 4.3: Contracts and definition files.



Source: The authors

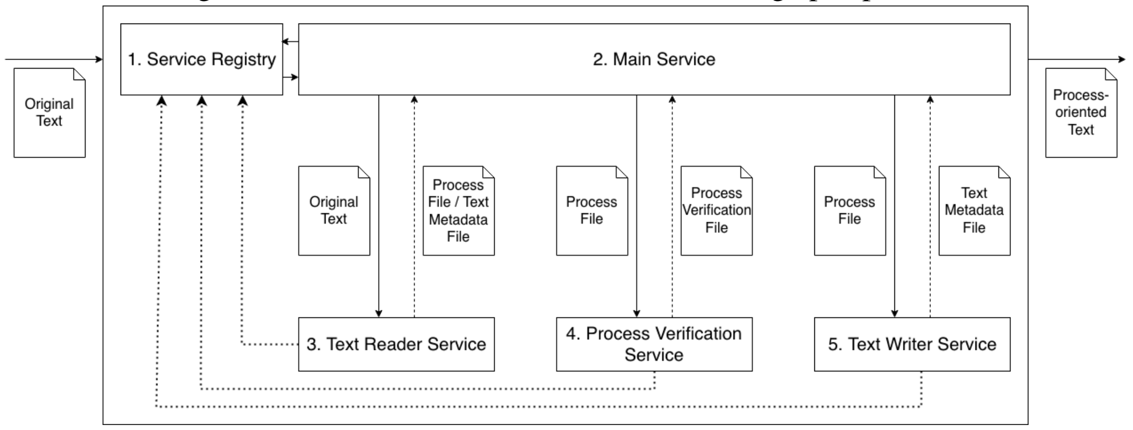
Definition, *Text Writer Definition*, and *Process Verification Definition*. Figure 4.3 presents a representation of the *data definitions files* and the *service definitions files*, as well as how they relate to the contracts created. In the following sections, the *data definition files* and *service definition files* will be presented in more details.

4.1.1 File Exchanges

The architecture developed can still be seen from the perspective of the files that are sent and received by the services. Figure 4.4 presents services, what they expect as input, and what they are capable of producing as output. As can be seen, there are three possible files, being: *Process File*, *Text Metadata File*, and *Process Verification File*. These files will be defined respectively by the data definition files *Process File Schema*, *Text Metadata File Schema*, and *Process Verification File Schema*.

The *Process File* is responsible to represent the business process. In this sense, this file represents the process elements, as well as how these process elements are related to each other. This file is used at two different moments. In the first moment, the file

Figure 4.4: Architecture overview: File exchange perspective.



Source: The authors

is created by the *Text Reader Service* and describes the process elements and their relationships. In the second moment, the file is read by the *Text Writer Service* and *Process Verification Service*.

The *Text Metadata File* is responsible for returning the text produced and the markings related to the process. These markings act as a mapping between the text and the business process. In addition, they allow to mark process elements in the text and make changes in the text structure. As an example, consider the sentence described in Figure 4.5. In order to mark the activities present in this sentence, it is necessary to know if there is a snippet of the sentence (i.e., part of the sentence) that describes a process element of type *activity* (i.e., process element type), the position in the sentence where this snippet starts (i.e., start index), and the position in the sentence where this snippet ends (i.e., end index).

Figure 4.5: Example of sentence with metadata.

$$\text{The process starts when } \underbrace{\overset{\substack{\text{start index}\\ \downarrow}}{\text{t}}\text{he technician performs an evaluatio}\overset{\substack{\text{end index}\\ \downarrow}}{\text{n}}}_{\text{process element type}}.$$

Source: The authors

The *Process Verification File* is responsible for returning the process issues observed. In this file, all issues identified in the process are listed along with a reference to the process element in which the problem was encountered. This file is used at two different moments. In the first moment, the file is created by the *Process Verification Service* to describe the issues identified. In the second moment, this file is used by the *Main Service* in generation of the business process-oriented text.

4.1.2 Behavior of Services for the Generation of Business Process-oriented Texts

For the generation of business process-oriented texts, the *Service Registry* is initially started to receive the new services. Then, service providers (i.e., *Text Reader Service*, *Process Verification Service*, and *Text Writer Service*) are registered in the *Service Registry*. Subsequently, the *Main Service* is started and is waiting for a user request.

When the *Main Service* receives a text as input (i.e., typed text or file in TXT format), it searches the *Service Registry* for the location of a service that performs text reading. The *Service Registry* then returns the location of the *Text Reader Service* to the *Main Service*. Subsequently, the *Main Service* communicates with the *Text Reader Service* and forwards the natural language text. The *Text Reader Service* then processes the natural language text and returns the identified process elements as well as their relationships (i.e., *Process File*) to the *Main Service*.

After receiving the *Process File* from the *Text Reader Service*, the *Main Service* searches the *Service Registry* for the location of a service that performs business process verification. Then, the *Service Registry* returns the location of the *Process Verification Service* to the *Main Service*. Subsequently, the *Main Service* communicates with the *Process Verification Service* and forwards the *Process File*. The *Process Verification Service* then processes the business process and returns the process issues that were identified (in relation to BPMN 2.0 and soundness) to the *Main Service* as a *Process Verification File*.

In addition, the *Main Service* searches the *Service Registry* for the location of a service that performs the generation of text with metadata. Then, the *Service Registry* returns the location of the *Text Writer Service* to the *Main Service*. Subsequently, the *Main Service* communicates with the *Text Writer Service* and forwards the *Process File*. The *Text Writer Service* then processes the business process and returns the text with metadata to the *Main Service* as a *Text Metadata File*.

Having received the *Process Verification File* and *Text Metadata File*, the *Main Service* generates a web page and sends it to the user so that it can access and interact with the business process-oriented text.

As defined in this section, it is possible that *Text Reader Service* also produces metadata for the original text. In this case, the *Main Service* would send the original text to the *Text Reader Service* and this, in turn, would send in response a file of type *Text*

Metadata File.

4.2 Data Definition Files

To perform the communication between services, it is necessary to define the data that will be shared between them. The files used to define the data that will be shared are the *data definition files*. In the context of this work, three *data definition files* were defined for the communication between the services, namely: *Process File Schema*, *Text Metadata File Schema*, and *Process Verification File Schema*. These files have been defined as XSD files since it is a common file format to describe this type of document (ERL, 2008).

Data definition files of this work have been converted automatically from XSD files to Java classes through the Java Architecture for XML Binding (JAXB) API (FIALLI; VAJJHALA, 2003). Thus, it is possible to manipulate (i.e., insert, modify, or delete) the data that will compose the files that will be shared between the services working directly on the JAVA attributes and classes. In the following sections each of the *data definition files* will be defined and detailed.

4.2.1 Process File Schema

The *Process File Schema* will be the file responsible for defining how the *Process File* should be. To represent the process elements as well as their relationships the BPMN file format was used as *Process File*.

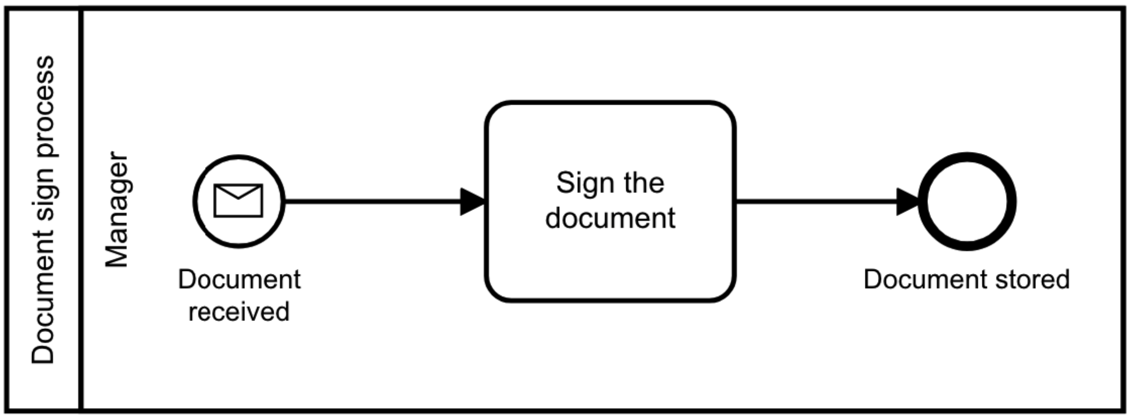
As discussed previously, the BPMN file format is a standard maintained by OMG for representing processes following the BPMN 2.0 (OMG, 2013). The BPMN file format allows to describe business processes in a file in XML format. The files used to define and generate the BPMN file format were taken from the OMG site[1], which are: BPMN20.xsd, BPMNDI.xsd, DC.xsd, DI.xsd, Semantic.xsd. Therefore, the *Process File Schema* is actually a set of these five XSD files provided by the OMG.

One of the advantages of using the BPMN file format is that since this is the default BPMN file to describe the business process models, well-known and largerly used

[1]https://www.omg.org/spec/BPMN/2.0/About-BPMN; last acessed 2019-02-02

BPMN modelling tools (e.g., Bizagi[2], Bonita[3], Signavio[4], Bpmn.io[5]) can export their process models to this format. Consequently, not only texts, but also process models can be directly provided as inputs for the generation of business process-oriented texts. In addition, using the BPMN file makes it easy to update the approach to new versions of BPMN specification. This is possible because, to ensure that it conforms to the current version, it is only necessary to get the new version of the files provided by OMG, automatically generate classes using JAXB and, if necessary, update the service.

Figure 4.6: BPMN example: Document sign.



Source: The authors

According to OMG (2013), the BPMN file seeks to cover two different aspects in a process: *process models* and *process diagrams*. *Process models* contains the semantics of the model. This aspect describes what the process elements are and how they are related. As an example, the process model represented in Figure 4.6 has 7 process elements (i.e., 1 pool, 1 lane, 2 events, 1 task, and 2 sequence flows). Among these elements, it is possible to note input and output relationships between certain flow objects and sequence flows (e.g., the start event has a sequence flow as output). On the other hand, *process diagrams* store the visual representation of business processes. This aspect seeks to cover visual questions such as the size and positioning of process elements in the model OMG (2013). Considering Figure 4.6, *process diagrams* is related to information, such as: pool size, lane size, and positioning of process elements.

Listing 4.1 presents the BPMN file for the process model depicted in Figure 4.6. As can be seen, the file starts with a header addressing the location of the tags (line 2). Next, the *bpmn:collaboration* (lines 3 to 5) is informed. In this step, all participants and

[2]https://www.bizagi.com; last acessed 2019-02-02
[3]https://www.bonitasoft.com; last acessed 2019-02-02
[4]https://www.signavio.com; last acessed 2019-02-02
[5]https://bpmn.io; last acessed 2019-02-02

message flows represented are listed.

In sequence, the file describes the *bpmn:process* (lines 6 to 27). This part of the file is responsible for describing the lanes (lines 7 to 13) of the process, the process elements that belong to each lane (e.g, lines 9 to 11), the *flow objects* (lines 14 to 24) and *sequence flows* (lines 25 to 26) with their respective properties (e.g., id, name, incomings, outgoings, sourceRef, targetRef). As an example, the process element shown in the lines 18 to 21 is a task (i.e., *bpmn:task*) with id "T_1" and label "Sign the document". In addition, this task has as input the process element of id "Sf_1" and as output the process element of id "Sf_2". These process elements, in turn, are two sequence flows (i.e., *bpmn:sequenceFlow*) described in the lines 25 and 26 respectively. A file in BPMN format can describe multiple pools. In this case, there will be multiple *bpmn:process* in the BPMN file.

Finally, in the *bpmndi:BPMNDiagram* (line 28 to 60) is where the features present in the *process diagram aspect* are described. In this part of the document, the position and size of the process elements presented in *bpmn:process* (lines 6 to 27) are defined. As an example, the process element presented in lines 18 to 21 has a graphic representation in the lines 42 to 44. This process element will be represented in the process model at position 299 of axis X and 98 of axis Y. In addition, this process element will have width 100 and height of 80.

Listing 4.1: BPMN file example: Document sign.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bpmn:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
    " xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL"
    xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI" xmlns:dc="
    http://www.omg.org/spec/DD/20100524/DC" xmlns:di="http://www.omg.
    org/spec/DD/20100524/DI" id="Definitions_0hf160f" targetNamespace="
    http://bpmn.io/schema/bpmn">
3   <bpmn:collaboration id="Collaboration_1">
4     <bpmn:participant id="Par_1" name="Document sign process"
        processRef="P_1" />
5   </bpmn:collaboration>
6   <bpmn:process id="P_1" isExecutable="false">
7     <bpmn:laneSet id="Ls_1">
8       <bpmn:lane id="L_1" name="Manager">
9         <bpmn:flowNodeRef>Se_1</bpmn:flowNodeRef>
10        <bpmn:flowNodeRef>T_1</bpmn:flowNodeRef>
11        <bpmn:flowNodeRef>Ee_1</bpmn:flowNodeRef>
```

```
      </bpmn:lane>
    </bpmn:laneSet>
    <bpmn:startEvent id="Se_1" name="Document received">
      <bpmn:outgoing>Sf_1</bpmn:outgoing>
      <bpmn:messageEventDefinition />
    </bpmn:startEvent>
    <bpmn:task id="T_1" name="Sign the document">
      <bpmn:incoming>Sf_1</bpmn:incoming>
      <bpmn:outgoing>Sf_2</bpmn:outgoing>
    </bpmn:task>
    <bpmn:endEvent id="Ee_1" name="Document stored">
      <bpmn:incoming>Sf_2</bpmn:incoming>
    </bpmn:endEvent>
    <bpmn:sequenceFlow id="Sf_1" sourceRef="Se_1" targetRef="T_1" />
    <bpmn:sequenceFlow id="Sf_2" sourceRef="T_1" targetRef="Ee_1" />
  </bpmn:process>
  <bpmndi:BPMNDiagram id="BPMNDiagram_1">
    <bpmndi:BPMNPlane id="BPMNPL_1" bpmnElement="Collaboration_1">
      <bpmndi:BPMNShape id="Par_1_di" bpmnElement="Par_1">
        <dc:Bounds x="105" y="60" width="451" height="163" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="L_1_di" bpmnElement="L_1">
        <dc:Bounds x="135" y="60" width="421" height="163" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Se_1_di" bpmnElement="Se_1">
        <dc:Bounds x="181" y="120" width="36" height="36" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="174" y="163" width="51" height="40" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="T_1_di" bpmnElement="T_1">
        <dc:Bounds x="299" y="98" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Ee_1_di" bpmnElement="Ee_1">
        <dc:Bounds x="475" y="120" width="36" height="36" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="451" y="163" width="84" height="14" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge id="Sf_1_di" bpmnElement="Sf_1">
        <di:waypoint x="217" y="138" />
        <di:waypoint x="299" y="138" />
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="Sf_2_di" bpmnElement="Sf_2">
        <di:waypoint x="399" y="138" />
        <di:waypoint x="475" y="138" />
      </bpmndi:BPMNEdge>
    </bpmndi:BPMNPlane>
  </bpmndi:BPMNDiagram>
</bpmn:definitions>
```

As the objective of this work is in the identification of process elements and in the relationship between them, it will be focused only on the *process model aspect*. In this context, this work focused on the creation and reading of tags belonging to the *bpmn:collaboration* and *bpmn:process*.

4.2.2 Process Verification File Schema

For the representation of the *Process Verification File*, an XSD file was created. Listing 4.2 depicts this XSD file named *Process Verification File Schema*. The first element responsible for including all the others is called *messageList* (line 9). The *messageList* tag is composed of 0 or more verification messages called *message* (line 13). Each *message* is composed of four elements, being them:

- **description:** Displays the description of the verification (line 15).
- **processElementId:** Represents the index of the process element. This index helps to map the process elements to those described in the *Text Metadata File*. In addition, it can assume the value of "Process" in situations where the error is not related specifically to a process element (line 17).
- **messageType:** Displays the type of verification message. Any string can be used as value of this tag. However, taking into account the current *Process Verification Service*, the possible values currently assigned to this variable are: structure, label, and pragmatic (line 18).
- **source:** Displays where the message was generated from. Any string can be used as value of this tag. However, taking into account the current *Process Verification Service*, the possible values currently assigned to this variable are: BPMN verification,

YAWL verification (line 19).

Listing 4.2: Process verification file schema.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
3             xmlns:ve="http://br/edu/ufrgs/inf/bpm/verificationmessages"
4             targetNamespace="http://br/edu/ufrgs/inf/bpm/
                  verificationmessages">
5
6      <xs:element name="bpmnVerification" type="ve:tBpmnVerification"/>
7      <xs:complexType name="tBpmnVerification">
8          <xs:sequence>
9              <xs:element name="messageList" type="ve:tMessage"
                   maxOccurs="unbounded"/>
10         </xs:sequence>
11     </xs:complexType>
12     <xs:element name="message" type="ve:tMessage"/>
13     <xs:complexType name="tMessage">
14         <xs:all>
15             <xs:element name="description" type="xs:string"/>
16         </xs:all>
17         <xs:attribute name="processElementId" type="xs:string"/>
18         <xs:attribute name="messageType" type="xs:string"/>
19         <xs:attribute name="source" type="xs:string"/>
20     </xs:complexType>
21 </xs:schema>
```

For the *Process Verification Files*, JSON was used as the data interchange format. This format was used because it is faster and uses fewer resources than XML files (NURSEITOV et al., 2009). An example of *Process Verification File* produced by the *Process Verification Service* will be presented in the following sections.

4.2.3 Text Metadata File Schema

For the representation of the *Text Metadata File*, an XSD file was created. Listing 4.3 depicts this XSD file named *Text Metadata File Schema*. The first tag responsible for including all the others is called *textMetadata* (line 6). A *textMetadata* is composed of two elements, being them:

- **processList:** List that contains all the processes present in *textMetadata*. (line 9).
- **text:** A textual description with metadata. (line 10).

A *processList* can contain zero or more tags of type *process* (line 13). A *process* represents a participant (i.e., a pool in the BPMN 2.0) and is composed of three elements, being them:

- **resourceList:** List that contains all resources present in the process (line 16).
- **id:** The process identifier. In terms of the business process model, this information is related to the process id (line 18).
- **name:** The name of the process (line 19).

A *resourceList* can contain zero or more tags of type *resource* (line 21). A *resource* represents a business role (i.e., a lane in the BPMN 2.0) and is composed of two elements, being them:

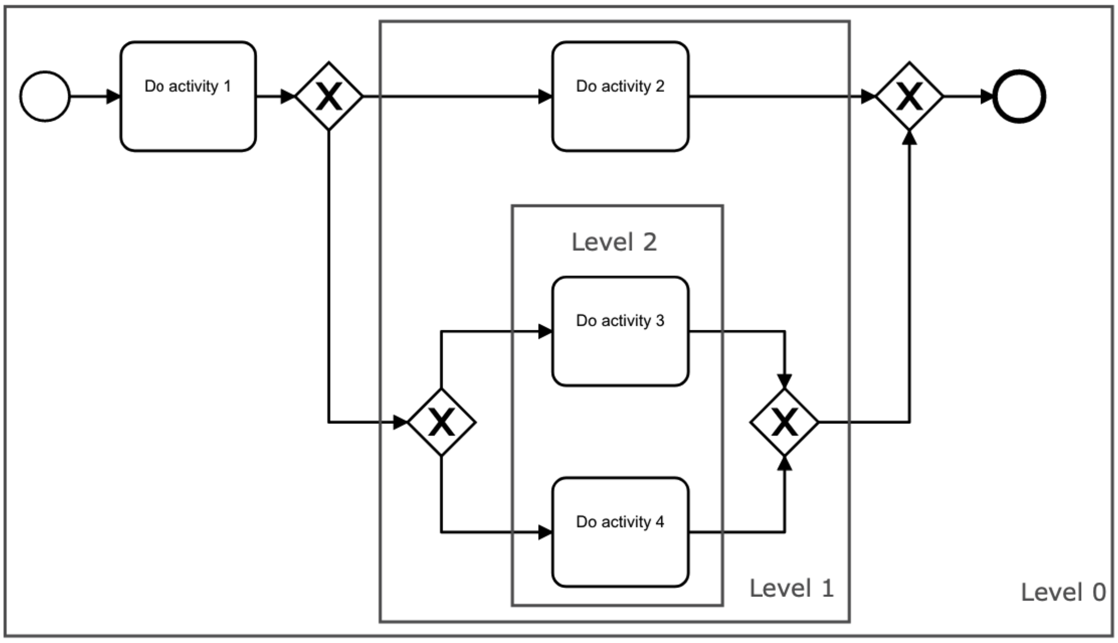
- **id:** The resource identifier. In terms of the business process model, this information is related to the lane id (line 23).
- **name:** The name of the resource (line 24).

A *text* can contain zero or more tags of type *sentence* (line 32). Each sentence is responsible for representing a sentence created in the text. The *sentence* is composed of three elements, being them:

- **value:** Represents the sentence that will be used in the text (line 35).
- **snippetList:** List containing information about snippets from the sentence (line 36).
- **newSplitPath:** Attribute that stores a Boolean value that represents whether the sentence is a *new split path*. In the context of this work, a sentence will be considered a *new split path* if it describes a process element presented immediately after a split gateway (i.e., a new path generated by a split gateway) (line 38).

A *snippetList* can contain zero or more tags of type *snippet* (line 40). A tag *snippet* presents data referring to a process element in a given part of a sentence. The *snippet* is composed of seven elements, being them:

Figure 4.7: Levels in a process model.



Source: The authors

- **startIndex:** Index of where the sentence snippet starts (line 42). If it starts at the beginning of the sentence, the value assigned to the *startIndex* will be "0". In addition, escape characters will not be considered. Thus, in the sentence "If \"condition\", do the activity 1.", if it is necessary to highlight the word "condition" the *startIndex* should be in the letter "c" which, in turn, has the index "4" (i.e., "I" = index 0, "f" = index 1, " " = index 2, "\"" = index 3, and "c" = index 4).

- **endIndex:** Index of where the sentence snippet ends (line 43). As an example, in the sentence "If \"condition\", do the activity 1.", if it is necessary to highlight the word "condition" the *endindex* should be the last letter "n" in the word which, in turn, has the index "12".

- **processElementId:** Represents the index of the process element. This index helps map the process elements described in the text to the original business process. In addition, it helps in mapping the process elements with the *Process Verification File* (line 44).

- **processElementType:** Represents the process element described in the snippet. This attribute helps in marking the process elements in the text (line 45).

- **resourceId:** Presents the id of the resource to which the process element described by the snippet is associated (line 46).

- **processId:** Presents id of the process to which the process element described by the snippet is associated (line 47).

- **level:** Attribute that stores an integer referring to the level that the snippet is in relation to the business process (line 48). Just as Leopold, Mendling and Polyvyanyy (2014) do to indent the different sentences produced by their approach, when the sentence describes elements that are not internal to gateways, this will have the level with value "0". However, if the sentence describes an element internal to a gateway, it will receive the level with value "1". This value will be incremented as the gateways are nested (see Figure 4.7).

Listing 4.3: Text metadata file schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:mt="http://br/edu/ufrgs/inf/bpm/textMetadata"
           targetNamespace="http://br/edu/ufrgs/inf/bpm/textMetadata">

    <xs:element name="textMetadata" type="mt:ttextMetadata"/>
    <xs:complexType name="tTextMetadata">
        <xs:sequence>
            <xs:element name="processList" type="mt:tProcess"
               minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="text" type="mt:tText"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="process" type="mt:tProcess"/>
    <xs:complexType name="tProcess">
        <xs:sequence>
            <xs:element name="resourceList" type="mt:tResource"
               minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:string"/>
        <xs:attribute name="name" type="xs:string"/>
    </xs:complexType>
    <xs:element name="resource" type="mt:tResource"/>
    <xs:complexType name="tResource">
        <xs:attribute name="id" type="xs:string"/>
        <xs:attribute name="name" type="xs:string"/>
    </xs:complexType>
    <xs:element name="text" type="mt:tText"/>
    <xs:complexType name="tText">
        <xs:sequence>
            <xs:element name="sentenceList" type="mt:tSentence"
               maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="sentence" type="mt:tSentence"/>
    <xs:complexType name="tSentence">
        <xs:sequence>
            <xs:element name="value" type="xs:string"/>
            <xs:element name="snippetList" type="mt:tSnippet"
               minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="newSplitPath" type="xs:boolean"/>
    </xs:complexType>
    <xs:element name="snippet" type="mt:tSnippet"/>
    <xs:complexType name="tSnippet">
        <xs:attribute name="startIndex" type="xs:int"/>
        <xs:attribute name="endIndex" type="xs:int"/>
        <xs:attribute name="processElementId" type="xs:string"/>
        <xs:attribute name="processElementType" type="xs:string"/>
        <xs:attribute name="resourceId" type="xs:string"/>
        <xs:attribute name="processId" type="xs:string"/>
        <xs:attribute name="level" type="xs:int"/>
    </xs:complexType>
</xs:schema>
```

The *processElementType* attribute was represented as a string and not as a closed set of elements for two reasons. First, it was not intended to be limited to the process elements allowed by BPMN 2.0 (i.e., the same as those which compose the *Process File*) since this could limit the communication between the services. As an example, BPMN 2.0 has a process element called *ExclusiveGateway* (i.e., XOR gateway). In order to know if this gateway is of type split or of type join it is necessary to realize a processing verifying the amount of incomings and outgoings that gateway owns. However, this information does not exist in the *Text Metadata File* so that either it would have to be added or it would not be possible to distinguish using only the data provided by this file. Thus, having its own format to describe the types of process elements allows greater expression. In this case, it could be provided as a “XORSPLIT” value to represent an XOR-split gateway and “XORJOIN” value to represent an XOR-join gateway. The second reason is to avoid limiting the possibilities of types of process elements even in custom sets of process elements. Since a string is not a closed set of possibilities, new services will not be limited to an initially defined set. In addition, if it were a custom set of process elements, these elements would need to be described in the *Text Metadata File Schema*. Thus, the *Text Metadata File Schema* would need to be updated whenever it was to support new process elements.

As well as *Process Verification File*, the *Text Metadata File* also uses JSON as the data interchange format. An example of *Text Metadata File* produced by the *Text Writer Service* will be presented in the following sections.

4.3 Service Definition Files

To perform the communication between services, it is necessary to define the interfaces between them in terms of possible requests and responses that a service can meet. The files used to represent the interfaces of a service are the *service definition files*. In the context of this work, were defined three *service definition files* for the communication between the services, namely: *Text Reader Definition*, *Process Verification Definition*, and *Text Writer Definition*. These files have been described as WADL files (HADLEY, 2006). In addition to being a file format that is often used to describe service definitions in REST services, the WADL has been chosen because it allows to use XSD files as *Data Definition Files*, which is desirable since *Process File Schema* uses XSD files (i.e., BPMN20.xsd, BPMNDI.xsd, DC.xsd, DI.xsd, and Semantic.xsd). Moreover, WADL files can be automatically converted to JAVA classes through the Apache CXF (BALANI; HATHI, 2009).

As an example of *service definition file*, Listing 4.4 presents the *Text Reader Definition*. This document is composed of two parts: *grammars* (lines 4 to 7) and *resources* (lines 8 to 35). *Grammars* include the references of the *data definition files* used in this service. In the context of the *Text Reader Service*, the *data definitions files* are, respectively, for the generation of processes through the *Process File* (BPMN20.xsd in line 5) and for the generation of text with metadata through the *Text Metadata File* (TextMetadata.xsd in line 6).

Resources, on the other hand, represents the features that are made available by the service. The *Text Reader Definition* has two features, being them: *generateProcess* (lines 10 to 21) and *generateText* (lines 22 to 33). The feature *generateProcess* receives as request a natural language text (line 11) and produces as response a *Process File* (i.e., represented as media type *application/xml* in line 18). The feature *generateText*, in turn, receives as request a natural language text (line 23) and produces as response a *Text Metadata File* (i.e., represented as media type *application/json* in line 30).

Listing 4.4: Text reader definition.

```
 1 <?xml version="1.0"?>
 2 <application xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 3              xmlns="http://wadl.dev.java.net/2009/02">
 4     <grammars>
 5         <include href="schemas/BPMN20.xsd"/>
 6         <include href="schemas/TextMetadata.xsd"/>
 7     </grammars>
 8     <resources>
 9         <resource path="/service" id="br.edu.ufrgs.inf.bpm.service.
              ITextReaderService">
10             <resource path="/generateProcess">
11                 <method name="POST" id="generateProcess">
12                     <request>
13                         <representation mediaType="application/x-www-
                              form-urlencoded">
14                             <param name="text" style="query" type="
                                  xsd:string"/>
15                         </representation>
16                     </request>
17                     <response>
18                         <representation mediaType="application/xml"/>
19                     </response>
20                 </method>
21             </resource>
22             <resource path="/generateText">
23                 <method name="POST" id="generateText">
24                     <request>
25                         <representation mediaType="application/x-www-
                              form-urlencoded">
26                             <param name="text" style="query" type="
                                  xsd:string"/>
27                         </representation>
28                     </request>
29                     <response>
30                         <representation mediaType="application/json"/>
31                     </response>
```

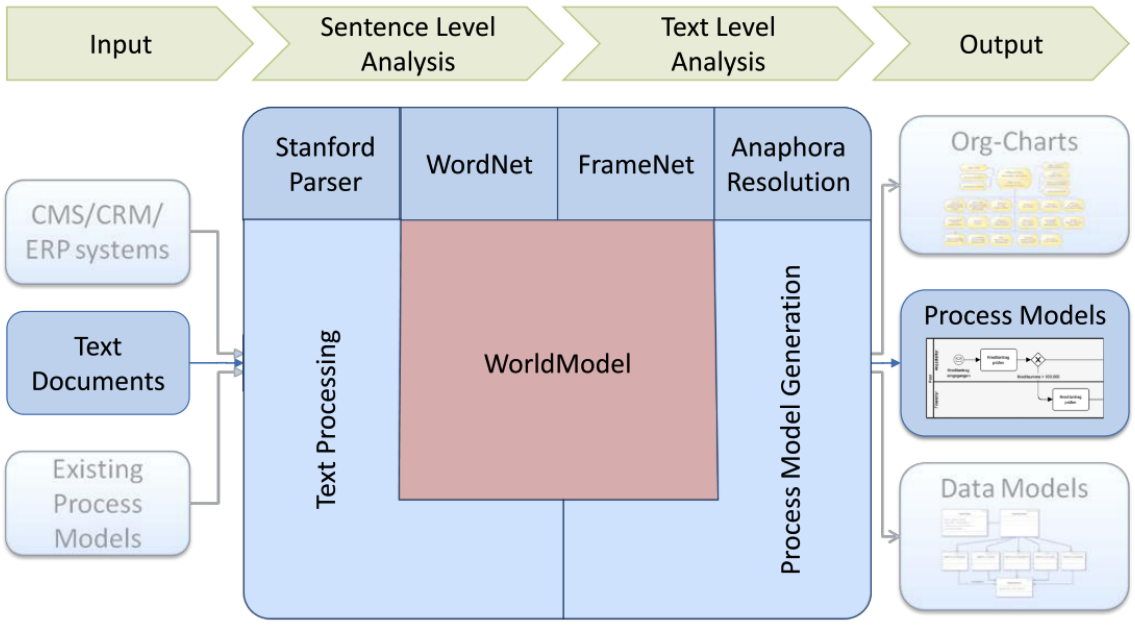
```
                </method>
            </resource>
        </resource>
    </resources>
</application>
```

4.4 Text Reader Service

The *Text Reader Service* aims to identify the process elements and their respective relationships in a business process textual description. This service provides two features. The first feature allows the service to receive as input a natural language text (i.e., *Original Text*) and produces as output a file containing the identified process elements as well as their relationships (i.e., *Process File*). The second functionality, in turn, also receives natural language text as input, but produces as output a text containing metadata (i.e., *Text Metadata File*).

For the construction of this service, the approach for process model generation from natural language text proposed by Friedrich, Mendling and Puhlmann (2011) was used. This approach performs an analysis of the textual description at the *sentence level* and at the *text level*. Figure 4.8 presents an overview of this approach.

Figure 4.8: Process model generation from natural language text overview.



Source: Friedrich (2010)

In the *sentence level* analysis, the approach proposed by Friedrich, Mendling and

Puhlmann (2011) seeks to identify the *actions* in the text. For this, the text was separated into individual sentences and the *Stanford Parser* (MARNEFFE et al., 2006) was used to extract the information present in the sentences. Then, these *actions* are stored in an intermediate structure called *World Model*.

In the *text level* analysis, the approach seeks to enrich the data stored in the *World Model*. At this point, the previously identified *actions* are related. For this, the authors make use of an *anaphora resolution technique* designed by them to identify words that are replaced in the text by pronouns (e.g., he, she, it) and determiners (e.g., this, that). Without the anaphora resolution technique, the sentences "(1) The *employee* signs the report. (2) Then *he* forwards the report to the manager." would be considered as activities performed by two different resources (i.e., *employee* and *he*) when in fact they are carried out by the same resource (i.e., *employee*). In addition, the authors seek to identify some markers present in the text that may indicate relationships between *actions* (e.g., "if", "in parallel", "otherwise"). Also, the authors make use of the *WordNet* (MILLER, 1995) and *FrameNet* (BAKER; FILLMORE; LOWE, 1998) lexical databases to perform an extraction of the meaning of words and phrases in the text and thus, perform the semantic analysis of the text.

The approach proposed by Friedrich, Mendling and Puhlmann (2011) was chosen as the basis for the *Text Reading Service* because it allows, from a description of the process, to find the process elements and how they are related. Moreover, this approach presents its own technique of anaphora resolution, which helps to find the resources that perform activities, even when these resources are represented in the text as pronouns or determiners. In addition, the approach was validated with 47 process descriptions and presented 76.98% similarity with the original process models. Finally, the approach is considered the state of the art by recent works (RIEFER; TERNIS; THALER, 2016).

4.4.1 Modifications in the Process Model Generation from Natural Language Text Approach

In relation to the modifications, the prototype built by Friedrich, Mendling and Puhlmann (2011) has been adapted to not only be a Java project and start behaving like a REST service. In order to do this, the *Text Reader Contract* has added to the prototype. In addition, Java code was created to define how to interact with the service (i.e., define how the inputs should be, how the outputs should be, how the service should behave for a given request).

To meet the first functionality of *representing the process elements and their relationships*, the business process produced by the approach was converted to a *Process File*. On the other hand, to meet the second functionality of *representing the text containing metadata*, the data extracted from the original text were collected so that, together with the original text, they were transformed into the *Text Metadata File*.

Since this approach deals with the original text and not with a controlled generated text, it becomes more complicated to define precisely which parts of a sentence present which process elements. Thus, for the *Text Reader Service* in the current approach, it was defined that the *snippets* would be equivalent to the size of the sentence. Hence, every *snippet* will have as *startIndex* the value 0 and as *endIndex* the length of the sentence.

In the following section, an example of *Process File* generated from the *Text Reader Service* is presented. Since *Text Writer Service* also produces a *Text Metadata File*, an example of *Text Metadata File* will be presented only when the *Text Writer Service* is described (Section 4.6).

4.4.2 Text Reader Output

Listing 4.5 presents the *Process File* produced from the process description of Figure 2.4. The produced *Process File* contains a resource named “technician” (line 5) and all the process elements that are associated with this resource (lines 6 to 15). In addition, it is possible to identify 6 activities, which are: “perform an evaluation” (line 18), “format the computer” (line 22), “replace the part” (line 26), “fill out the part replacement form” (line 30), “make no modification to the computer” (line 34), and “complete the repair form” (line 38). Moreover, the document also has a XOR-split (line 42), a XOR-join (line 48), a start event (line 54) and an end event (line 57). Finally, it is possible to identify the sequence flows that are connecting the flow elements shown above (lines 60 to 70).

Listing 4.5: Process file example: Computer repair.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
    xmlns:ns2="http://www.omg.org/spec/DD/20100524/DI" xmlns:ns4="http:
    //www.omg.org/spec/DD/20100524/DC" xmlns:ns3="http://www.omg.org/
    spec/BPMN/20100524/DI">
```

```
<process name="Pool" id="id-0">
    <laneSet name="Pool" id="id-1">
        <lane name="technician" id="id-2">
            <flowNodeRef>id-3</flowNodeRef>
            <flowNodeRef>id-4</flowNodeRef>
            <flowNodeRef>id-5</flowNodeRef>
            <flowNodeRef>id-6</flowNodeRef>
            <flowNodeRef>id-7</flowNodeRef>
            <flowNodeRef>id-8</flowNodeRef>
            <flowNodeRef>id-9</flowNodeRef>
            <flowNodeRef>id-10</flowNodeRef>
            <flowNodeRef>id-11</flowNodeRef>
            <flowNodeRef>id-12</flowNodeRef>
        </lane>
    </laneSet>
    <task name="perform an evaluation" id="id-3">
        <incoming>id-18</incoming>
        <outgoing>id-13</outgoing>
    </task>
    <task name="format the computer" id="id-4">
        <incoming>id-20</incoming>
        <outgoing>id-16</outgoing>
    </task>
    <task name="replace the part" id="id-5">
        <incoming>id-21</incoming>
        <outgoing>id-14</outgoing>
    </task>
    <task name="fill out the part replacement form" id="id-6">
        <incoming>id-14</incoming>
        <outgoing>id-17</outgoing>
    </task>
    <task name="make no modification to the computer" id="id-7">
        <incoming>id-22</incoming>
        <outgoing>id-15</outgoing>
    </task>
    <task name="complete the repair form" id="id-8">
        <incoming>id-23</incoming>
        <outgoing>id-19</outgoing>
    </task>
    <exclusiveGateway id="id-9">
        <incoming>id-13</incoming>
```

```
            <outgoing>id-20</outgoing>
            <outgoing>id-21</outgoing>
            <outgoing>id-22</outgoing>
        </exclusiveGateway>
        <exclusiveGateway id="id-10">
            <incoming>id-15</incoming>
            <incoming>id-16</incoming>
            <incoming>id-17</incoming>
            <outgoing>id-23</outgoing>
        </exclusiveGateway>
        <startEvent id="id-11">
            <outgoing>id-18</outgoing>
        </startEvent>
        <endEvent id="id-12">
            <incoming>id-19</incoming>
        </endEvent>
        <sequenceFlow sourceRef="id-3" targetRef="id-9" name="" id="id-13"/>
        <sequenceFlow sourceRef="id-5" targetRef="id-6" name="" id="id-14"/>
        <sequenceFlow sourceRef="id-7" targetRef="id-10" name="" id="id-15"/>
        <sequenceFlow sourceRef="id-4" targetRef="id-10" name="" id="id-16"/>
        <sequenceFlow sourceRef="id-6" targetRef="id-10" name="" id="id-17"/>
        <sequenceFlow sourceRef="id-11" targetRef="id-3" name="" id="id-18"/>
        <sequenceFlow sourceRef="id-8" targetRef="id-12" name="" id="id-19"/>
        <sequenceFlow sourceRef="id-9" targetRef="id-4" name="a software problem  " id="id-20"/>
        <sequenceFlow sourceRef="id-9" targetRef="id-5" name="a hardware problem  " id="id-21"/>
        <sequenceFlow sourceRef="id-9" targetRef="id-7" name="founds no problem " id="id-22"/>
        <sequenceFlow sourceRef="id-10" targetRef="id-8" name="" id="id-23"/>
    </process>
</definitions>
```

4.4.3 Process Identification Issues

Although the approach has 76.98% similarity and is considered the state of the art by other works, it presents points to be improved in the identification of process elements. Regarding this fact, we try to highlight the main points responsible for decreasing the similarity in the process identification. These points were presented by the authors (FRIEDRICH; MENDLING; PUHLMANN, 2011) and also evidenced in our approach. The points observed are: *noise*, *processing problems*, and *different levels of abstraction*.

Noise is defined as sentences that are not part of the process description, but detail the data objects used or add information that is out of the process. An example of *noise* can be observed in activity "make no modification to the computer" (line 34 present in Listing 4.5). In the process description of Figure 2.4, the sentence s_4 only informs that no action should be taken if no problem is found. However, the identification approach considers that there is an *action* in this sentence and consequently turns it into an *activity* in the business process. Another example of *noise* would be to add a contextualization information in the process description. If in the process description of Figure 2.4 was added at the beginning the sentence "Company X performs the repair of computers.", this sentence would only aim to contextualize the process being described. However, the current approach will consider that *perform computer repair* is an *action* being performed by the actor *Company X* and thus transform it into an *activity* in the business process.

In relation to *processing problems*, the approach makes use of NLP tools widely studied and used. However, these tools presented some undesirable results. According to the authors, *Stanford Parser* and *FrameNet* failed to classify some verbs into sentences. In addition, *WordNet* performed some analyses that were not expected. An example observed by the authors can be seen in the sentence "The customer wants to buy a house". Once the term *house* is used to represent an aristocratic family line (e.g., "The house of York"), the term was classified as an "social group" an identified as an *actor* rather than as a *resource*[6] , which leads to problems during the anaphora resolution stage.

Regarding *different levels of abstraction*, the authors comment that similarity between the original process model and the process model generated by the approach may be low when these processes are described at different levels of granularity. This means, for example, to have a very detailed process description (i.e., process model generated with several details) and an original process model more abstract. Unlike the other points

[6]Response expected by the authors. Has no relation to the "resource" present in the *Text Metadata File*.

mentioned, this type of issue more affects the result generated by the validation (i.e., similarity) than the process of identifying elements themselves. However, it is important to note in relation to the *different levels of abstraction* that the process identified by the approach will be as detailed as the process description from which it originated.

Since these problems are inherent in the process identification step (i.e., *Text Reader Service*), they only affect the generation of business process-oriented text from natural language texts. Thus, business process-oriented texts generated from process models are not impaired by the above points. In any case, such points influence the quality of the text produced since they can lead to an incorrect understanding of the process. Thus, future works can be created to solve these points in order to improve the accuracy of the current identification approach and, consequently, the text generation approach proposed in this work.

4.5 Process Verification Service

This service aims to perform a verification of the business process represented by the *Process File*. To do this, this service must receive as input a *Process File* and produce as a response a set of verification messages in the form of a *Process Verification File*. This service aims to check the quality of the process in relation to the BPMN 2.0 and in relation to soundness. Therefore, it is not expected of this service to make changes to the text produced by the approach. The service should only check the business process and produce the verification messages of the problems or possible points of improvement identified.

If the verification message is not directly related to a single process element, the verification message is considered to be in the process and the value “Process” is assigned to the *processElementId* attribute of the *Process Verification File*. In addition, considering that the generated process description will transform the process to have a single input and a single output, we have also chosen to assign the value “Process” to the *processElementId* in verification messages associated with *start events* and *end events*.

The verifications carried out were selected from three different reasons. Firstly, some verifications were selected because they directly interfere in the quality of the process description due to the lack of information presented at the input (i.e., defined as *label verifications*). An example of this type of verification would be an activity in which it is not known which is the resource that executes it. Second, some verifications were selected because they identify issues that interfere with the quality of the process, producing processes with execution problems (i.e., defined as *structure verifications*). An example of this type of verification would be to find in a process an activity that will never be executed. Finally, since the process description can be used during the discovery phase for the design of the process model, some verifications were defined in order to anticipate potential problems in the future process model (i.e., defined as *pragmatic verifications*). Although pragmatic questions are directly related to the process model, identifying some problems at the beginning can facilitate the construction of the process models and avoid some future corrections. An example of this type of verification would be to verify that the process has OR gateways (i.e., something that should be avoided whenever possible in a process model (MENDLING; REIJERS; AALST, 2010)).

The verifications performed by the *Verification Service* come from two sources: *BPMN Verification* and *YAWL Verification*. *BPMN Verification* seeks to perform process verifications by considering BPMN 2.0 itself. On the other hand, *YAWL Verification* converts the process to a YAWL model and performs structural and soundness verifications. The following sections will go into detail in each of these sources.

4.5.1 BPMN Verification

For the source *BPMN Verification* a process verification is performed taking into account the BPMN 2.0. The factors considered are related to the absence of labels in process elements and some pragmatic factors (MENDLING; REIJERS; AALST, 2010). For this step, we developed JAVA codes to verify the process described in the *Process File*. Whenever an undesirable condition was identified, a verification message was created and added to the *Process Verification File*. These verification messages, in turn, are composed of predefined texts and some regions that must be filled according to what is being described. The possible regions are:

- **@elementId:** Presents the ID of the process element (e.g., “(id: Task1)”).
- **@element:** Presents the name and the ID of the process element (e.g., “\“Do activity 1\” (id: Task1)”).
- **@elementType:** Presents the type of the process element (e.g., “Activity”, “Start event”).

- **@processName:** Presents the name of the process (e.g., “Computer repair”).
- **@amountFind:** Presents a number that represents the amount of elements find.
- **@amountLimit:** Presents a number that represents the number of elements that is the expected threshold.

The complete set of verification considered in the *BPMN Verification*, as well as their respective possible verification messages are presented below.

- $\mathbf{V_1}$**:** Verify process elements without labels (i.e., "No label was identified in the @elementType @elementId.").
- $\mathbf{V_2}$**:** Verify that the outputs of an XOR-split/OR-split gateway (i.e., conditions) have labels (i.e., "Not all condition labels were identified in the gateway @element.").
- $\mathbf{V_3}$**:** Verify if an activity is associated with a resource (i.e., “No resource was identified in the @elementType @element.").
- $\mathbf{V_4}$**:** Verify if a process element that is not a *start event* does not have an input process element (i.e., "No incoming element was identified in the @elementType @element.").
- $\mathbf{V_5}$**:** Verify if a process element that is not an *end event* does not have an output process element (i.e., "No outgoing element was identified in the @elementType @element.").
- $\mathbf{V_6}$**:** Verify the amount of start events.
  1. $\mathbf{V_{6.1}}$**:** Alert when no start event is identified (i.e., "No start event was identified in the process @processName.").
  2. $\mathbf{V_{6.2}}$**:** Alert when multiple start events are identified (i.e., "Multiple start events were identified in the process @processName.").
- $\mathbf{V_7}$**:** Verify the amount of end events.
  1. $\mathbf{V_{7.1}}$**:** Alert when no end event is identified (i.e., "No end event was identified in the process @processName.").
  2. $\mathbf{V_{7.2}}$**:** Alert when multiple end events are identified (i.e., "Multiple end events were identified in the process @processName.").

- $\mathbf{V_8}$**:** Verify if the amount of process elements is greater than 30 (i.e., "The amount of elements in the process @element exceeded the maximum expected quantity (found @amountFind and is expected at most @amountLimit)").

- $\mathbf{V_9}$**:** Verify the presence of OR gateways in the process (i.e., "The element @element is a inclusive gateway. It is a good practice to avoid using this element in a process.").

4.5.2 YAWL Verification

In order to verify the soundness in the business process, the YAWL modeling language was used. In this sense, the approach proposed by Jianhong and Song (2010) to transform process models in the BPMN 2.0 format for YAWL networks was used. In relation to this transformation, all the process elements present in the scope of this work are properly mapped, except pools and lanes that do not present equivalents in the YAWL.

According to Adams and Hofstede (2016), the verification performed by YAWL can be separated into *validation* and *analysis*. In *validation*, its possible to validate the specification against YAWL syntax and semantics. Some of the problems that *validation* allows to identify are *check if the process element does not have an input*, *check if the process element does not have an output*, *check if the process element does not have to default outgoing flow*. However, given the transformation of BPMN 2.0 to YAWL, these errors are not even possible.

On the other hand, *analysis* allows a thorough analysis of the specification for identifying deadlocks and other issues. Moreover, it is only possible to perform an *analysis* when the *validation* presents no problem. The verification performed by YAWL allows you to identify various problems. Some of the key verifications as well as their respective verification messages can be observed below. For these cases, @*elementList* will be a list of @*element* usually separated by commas.

- $\mathbf{V_{10}}$**:** Check if the process has an option to complete (i.e., “The process does not have an option to complete. Potential deadlocks: @elementList”).

- $\mathbf{V_{11}}$**:** Find activities that will never be executed (i.e., “The process has unreachable elements: @elementList”).

- $\mathbf{V_{12}}$**:** Check if the process satisfy the soundness property (i.e., “The process does not satisfy the soundness property.”).

- $\mathbf{V_{13}}$**:** Check livelocks problems (i.e., “The element @element plays a part in an infinite loop/recursion in which no work items may be created.”).

In order to perform the *YAWL Verification*, the *Process File* is converted in a YAWL network using the approach proposed by Jianhong and Song (2010). Then, it is submitted to the YAWL tool to perform the *validation*. If problems are encountered, they are inserted into a list of verification messages. However, if no problems are found during the *validation*, the YAWL network is submitted to the YAWL tool to perform the *analysis*. If problems are found during the *analysis*, they are inserted into a list of verification messages. Finally, the verification messages are partially transformed to be more related to BPMN 2.0 and, therefore, more suitable to the process descriptions being created.

If errors are found during the *validation* step that make the *analysis* step impossible, the following message is also created: "It was not possible to verify soundness in the model. Please correct the errors identified in the YAWL before this is possible.".

4.5.3 Classifications of the Verifications

Following the *Process Verification File* presented in the Section 4.2.2, each verification was classified as coming from a particular source (e.g., BPMN Verification and YAWL Verification) and belonging to a particular message type (e.g., Label, Structure, and Pragmatic). Table 4.1 presents each verification (V_{ID}) with their respective source and message type.

4.5.4 Verification Service Output

Listing 4.6 presents the *Process Verification File* in JSON format generated by *Process Verification Service* for the BPMN 2.0 process model example represented in Figure 4.9. As defined by the *Process Verification File Schema*, each message will contain a description (i.e., *description*), a process element ID (i.e.,*processElementId*), a message type (i.e., *messageType*), and a source (i.e., *source*). As can be observed, 12 problems were found being 10 generated from the *BPMN Verification* (lines 3 to 62) and 2 generated from the *YAWL Verification* (lines 63 to 74). In addition, it is possible to define that

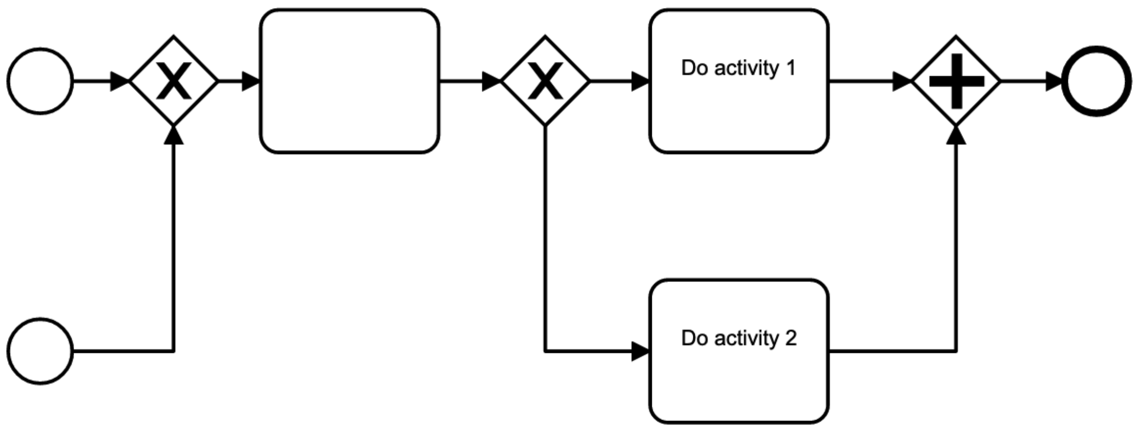
Table 4.1: Verification types classification.

V_{ID}	Source		Message Type		
	BPMN Verification	YAWL Verification	Label	Structure	Pragmatic
V_1	X		X		
V_2	X		X		
V_3	X		X		
V_4	X			X	
V_5	X			X	
V_6	X				X
V_7	X				X
V_8	X				X
V_9	X				X
V_{10}		X		X	
V_{11}		X		X	
V_{12}		X		X	
V_{13}		X		X	

Source: The authors

there is 1 message of type pragmatic (line 6), 9 messages of type label (lines 12, 18, 24, 30, 36, 42, 48, 54, and 60), and 2 messages of type structure (lines 66 and 72).

Figure 4.9: Process model with problems.



Source: The authors

Another thing that can be observed is the process elements that have the *processElementId* with a value "Process". They occur in cases where it is not possible to associate the verification message with a process element (lines 65 and 71), and in cases where the process element is a *start even*t or *end event* (lines 5, 11, 23, and 41). Finally, a highlight should be given for the description of the eleventh message (line 64). This message informs the user that the process does not have an option to complete and points to potential deadlocks. Although this is an error message generated by *YAWL Verification*, the elements have been modified for the BPMN 2.0 process elements.

Listing 4.6: Process verification file example: Process model with problems.

```
{
  "messageList": [
    {
      "description": "Multiple start events were identified in the
          process (id: Process_05u14rw).",
      "processElementId": "Process",
      "messageType": "pragmatic",
      "source": "BPMN Verification"
    },
    {
      "description": "No label was identified in the Start Event (id:
          StartEvent_1ai55ks).",
      "processElementId": "Process",
      "messageType": "label",
      "source": "BPMN Verification"
    },
    {
      "description": "No label was identified in the Activity (id:
          Task_1vmdkpu).",
      "processElementId": "Task_1vmdkpu",
      "messageType": "label",
      "source": "BPMN Verification"
    },
    {
      "description": "No label was identified in the Start Event (id:
          StartEvent_1whelzl).",
      "processElementId": "Process",
      "messageType": "label",
      "source": "BPMN Verification"
    },
    {
      "description": "No label was identified in the Exclusive Gateway
           (id: ExclusiveGateway_1fbl8y1).",
      "processElementId": "ExclusiveGateway_1fbl8y1",
      "messageType": "label",
      "source": "BPMN Verification"
    },
    {
      "description": "Not all condition labels were identified in the
          Exclusive Gateway (id: ExclusiveGateway_1fbl8y1).",
      "processElementId": "ExclusiveGateway_1fbl8y1",
      "messageType": "label",
      "source": "BPMN Verification"
    },
    {
      "description": "No label was identified in the End Event (id:
          EndEvent_12npddc).",
      "processElementId": "Process",
      "messageType": "label",
      "source": "BPMN Verification"
    },
    {
      "description": "No resource was identified in the Activity (id:
          Task_1vmdkpu).",
      "processElementId": "Task_1vmdkpu",
      "messageType": "label",
      "source": "BPMN Verification"
    },
    {
      "description": "No resource was identified in the Activity \"Do
          activtiy 1\" (id: Task_0rd32o3).",
      "processElementId": "Task_0rd32o3",
      "messageType": "label",
      "source": "BPMN Verification"
    },
    {
      "description": "No resource was identified in the Activity \"Do
          activity 2\" (id: Task_1eeo3ow).",
      "processElementId": "Task_1eeo3ow",
      "messageType": "label",
      "source": "BPMN Verification"
    },
    {
      "description": "The process does not have an option to complete.
           Potential deadlocks: 1c{Exclusive Gateway (id:
          ExclusiveGateway_1fbl8y1)_Activity: Do activity 2 (id:
          Task_1eeo3ow)} 2c{Exclusive Gateway (id:
          ExclusiveGateway_1fbl8y1)_Activity: Do activtiy 1 (id:
          Task_0rd32o3)} 2c{Exclusive Gateway (id:
          ExclusiveGateway_1fbl8y1)_Activity: Do activity 2 (id:
          Task_1eeo3ow)} 1c{Exclusive Gateway (id:
        ExclusiveGateway_1fbl8y1)_Activity: Do activtiy 1 (id:
        Task_0rd32o3)}",
      "processElementId": "Process",
      "messageType": "structure",
      "source": "YAWL Verification"
    },
    {
      "description": "The process does not satisfy the soundness
        property.",
      "processElementId": "Process",
      "messageType": "structure",
      "source": "YAWL Verification"
    }
  ]
}
```

4.6 Text Writer Service

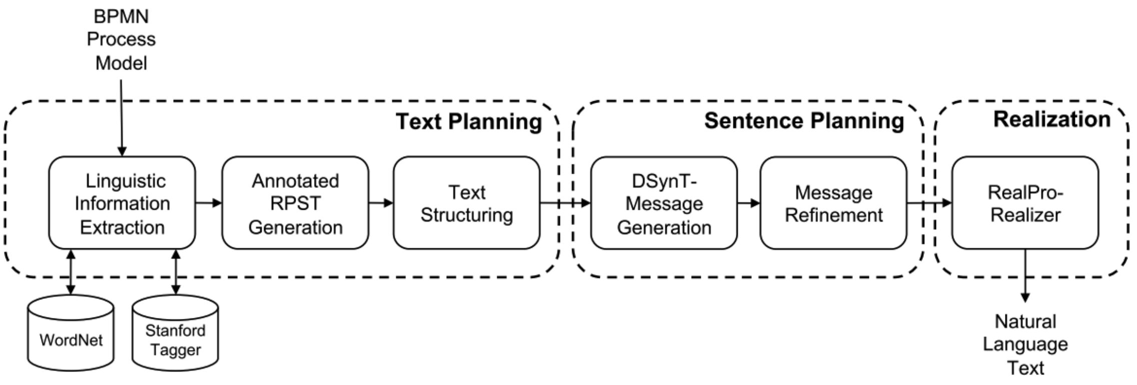
This service aims to generate a business process-oriented text from a process (i.e., process elements and their relationships). For this, this service should receive as input a *Process File* and output a business process description with metadata (i.e, *Text Metadata File*). These metadata will allow mark the process elements and structure the text.

For the construction of this service, it was used the approach of generating natural language texts from business process models proposed by Leopold, Mendling and Polyvyanyy (2012). The approach proposed by Leopold, Mendling and Polyvyanyy (2012) takes a process model as input and produces a natural language text as output. This approach consists of 6 different stages, separated into three phases, being: *Text Planning*, *Sentence Planning* and *Realization*. Figure 4.10 presents the architecture of the approach.

In the *Text Planning* phase is performed the extraction of linguistic components present in the labels of the process elements (i.e., *Linguistic Information Extraction stage*). In addition, it is performed the linearization of the process model through a Refined Process Structure Tree (RPST) (VANHATALO; VÖLZER; KOEHLER, 2009; POLYVYANYY; VANHATALO; VÖLZER, 2010) for every pool of the input model (i.e., *Annotated RPST Generation stage*).

An RPST is a parse tree that, from a process model, produces a tree containing a hierarchy of sub graphs called *fragments*. These *fragments* can be nested or disjoints

Figure 4.10: Original text generation approach architecture.



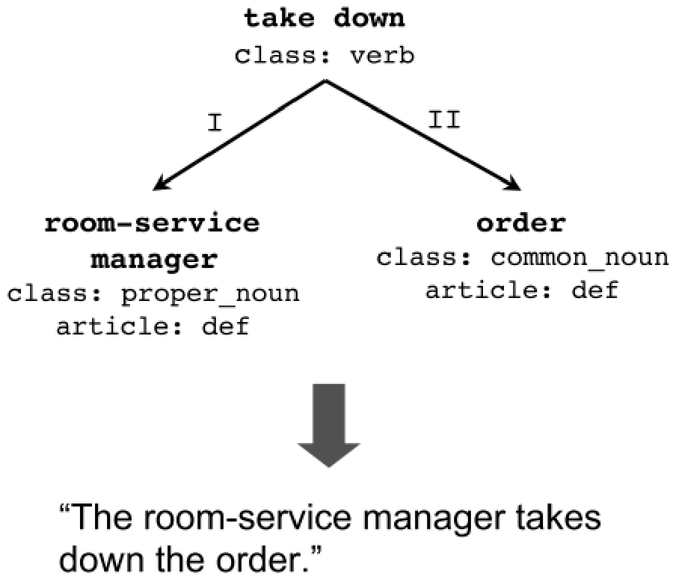
Source: Leopold, Mendling and Polyvyanyy (2014)

with each other and are classified into four categories, being: *trivial*, *bond*, *polygon*, and *rigid*. The *trivial* fragment represents two nodes connected with a single arc. This type of fragment will be used to represent the sequential link between two process elements (e.g., connection between the start event and the "Perform evaluation" activity in Figure 2.3). *Bond* fragment represents a set of fragments sharing two common nodes. This type of fragment will be used to represent process regions involved in the same split and join (e.g., the paths involved by XOR-split and XOR-join in Figure 2.3). *Polygon* fragment capture sequences of other fragments. Thus, when two or more fragments are presented sequentially, they make up a *polygon* fragment (e.g., the *trivial* fragment composed of the XOR-split and the "Replace part" activity and the *trivial* fragment composed of the "Replace part" activity and the "Fill the part replacement form" activity in Figure 2.3 together form a *polygon* fragment). Finally, when a fragment can not be classified as *trivial*, *bond* or *polygon*, it is classified as *rigid* fragment.

Moreover, in the *Text Planning* phase is performed the application of text structuring techniques, such as the insertion of paragraphs and bullet points (i.e., *Text Structuring stage*).

In the *Sentence Planning* phase, it is performed the generation of an intermediate message structure for each fragment of the RPST (i.e., *DSynT - Message Generation stage*). The intermediate message structure used by the authors is a Deep-Syntactic Tree (DSynT) (LAVOIE; RAMBOW, 1997). This structure represents the most significant aspects of the syntactic structure of a sentence and will be used as input by the *Realization* phase. Each node of a DSynT has a *lexeme*. Lexeme is a unit of lexicon that has a certain meaning. Lexemes appear in the texts as nouns, verbs, and adjective stems (BEARD; VOLPE, 2005). A lexeme may contain inflections and, thus, form a set of words known as inflected variations. As an example, the words *run*, *running*, *runs*, and *ran* are different variations from the same lexeme. Each lexeme can be complemented with meta information called *grammemes* able to bring characteristics of how the lexeme should present itself (e.g., the voice and tense of a verb). Moreover, the different lexemes that will compose the DSynT will be related through edges. An example of DSynT is show in Figure 4.11. As can be seen, the root node is represented by the main verb of the sentence. In addition, the relation I was used to represent the subject and the relation II to represent the object of the sentence.

Figure 4.11: DSynT example.



Source: Leopold, Mendling and Polyvyanyy (2014) (adapted)

In addition, in the *Sentence Planning* phase is also performed the refinement of messages (i.e., *Message Refinement stage*). The message refinement consists in the application of three techniques: *Message Aggregation*, *Referring Expressions*, and *Discourse Markers*. Regarding *Message Aggregation*, messages created in the previous stage can be combined to form new messages. As an example, the sentences "The technician must replace the part" and "The technician must fill out the part replacement form" could be aggregated into "The technician must replace the part *and* fill out the part replacement form". After the *Message Aggregation*, the *Referring Expressions* technique is performed. In this technique, adjacent messages that are executed by the same role had the role replaced by a pronoun from the second message. As an example, assuming that the above messages had not been aggregated, the second sentence would be transformed into "*He* must fill out the part replacement form". Finally, in the *Discourse Markers* technique, the authors identified messages that appear in sequence and assigned the connectors *then*, *afterwards*, and *subsequently* to them. As an example, assuming again that the above messages had not been aggregated, the second sentence would be transformed into "*Then*, he must fill out the part replacement form".

Finally, in the *Realization* phase is performed the transformation of the intermediate message structures into grammatically correct sentences (i.e., *RealPro Realizer stage*). This task can be performed by softwares called realizers. There are different realizers in the literature, such as: RealPro Realizer (LAVOIE; RAMBOW, 1997), TG/2 (BUSEMANN, 1996), Yet Another Generator (YAG) (MCROY; CHANNARUKUL; ALI, 2000), and SURGE (ELHADAD; ROBIN, 1997). In order to perform the realization, the authors used in their approach the software RealPro Realizer from CoGenTex (LAVOIE; RAMBOW, 1997). Among the factors that led them to choose this realizer are: the manageability of the intermediate structure, license costs, generation speed, and Java compatibility.

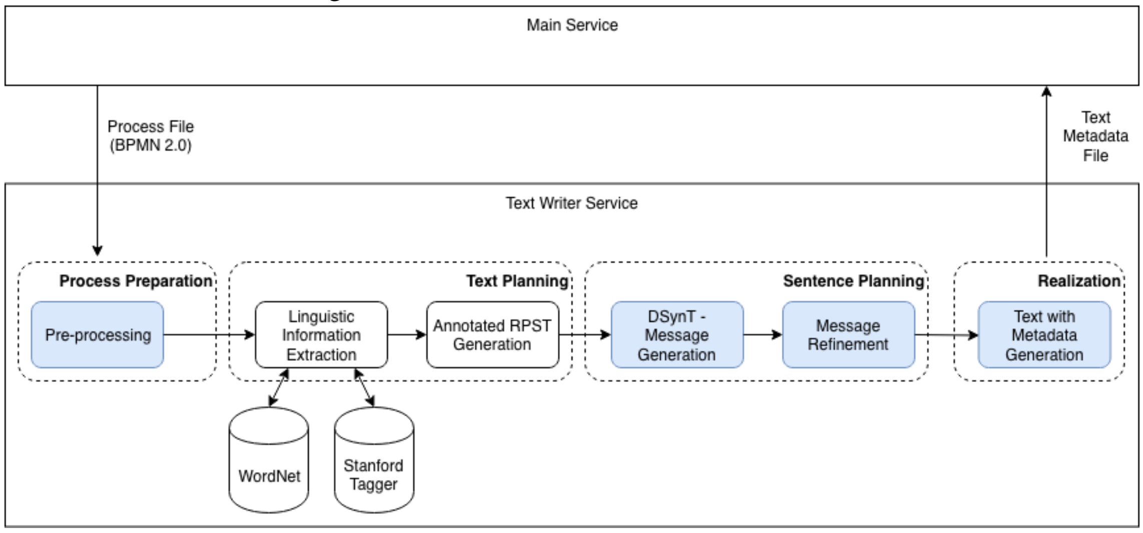
The approach proposed by Leopold, Mendling and Polyvyanyy (2012) was chosen for the creation of the *Text Writer Service* because it allows, from a BPMN 2.0 process model, to produce a natural language text. As the expected input of the *Text Writer Service* is a file in BPMN format (i.e., called *Process File*), the input in this approach is shown to be suitable for the service being constructed. In addition, the authors of the approach have validated it with users through a back translation technique where 11 students were asked to interpret process descriptions produced by the tool and try to reproduce the original process models. The results of the validation indicated that the students understand the text produced correctly, but they find it difficult to identify some process elements in a process description.

4.6.1 Modifications in the Generating Natural Language Texts from Business Process Models Approach

For the creation of the business process-oriented text defined in Chapter 3, this approach has received several modifications. Figure 4.12 shows the *Text Writer Service* presenting in detail the main modifications made in the original approach. Compared with the original approach, the following changes were made:

1. The input started to receive a *Pre-processing stage* prior to *Text Planning phase*, which verifies that the BPMN 2.0 process can be transformed in text and also applies small corrections to the process.
2. *Sentence Planning phase* has been modified to meet the requirements of the text

Figure 4.12: *Text Writer Service* overview.



Source: The authors

that was defined in Chapter 3.

3. The output is no longer a natural language text to become a *Text Metadata File*. This modification causes data to be sent about the process being described, which allows the marking of process elements in the text (e.g., highlight activities in the text) and the structuring of the text (e.g., paragraph indentation, paragraph split).

4. The *Text Structuring stage* was no longer the responsibility of the original approach and became the responsibility of the *Main Service*.

In the following subsections will be presented the main modifications made in the original approach in order to produce the *Text Writer Service*.

4.6.2 Pre-processing Phase

The *Pre-processing phase* has as objectives to verify if the process described in the *Process File* can be transformed in text and, if it can, to adjust the process before it is transformed. Unlike the *Process Verification Service* presented in Section 4.5, the verification performed in the *Pre-processing phase* only acts as a prerequisite for the generation of text by the *Text Writer Service*. This verification is carried out at two different moments.

At the first moment, it is checked whether the process described by the *Process File* contains all the sequence flows addressed to process elements. If it does not, the service returns an exception warning that there are sequence flows that are not connected and the generation of the text can not be performed.

In the second moment, the process is adjusted so that it can run by the tool. At this point, 3 checks are made: *pool labels verification*, *lane labels verification*, and *activity labels verification*. Regarding *pool labels verification*, the entire process is run in order to verify that all process elements are related to a pool. If there are elements that are not related, a generic pool called “Pool @id” where @id is an increment number is created for these elements. Regarding *lane labels verification*, it is checked whether the elements that belong to a given pool are associated with a lane. If not, it is created a lane per pool called “Resource @id” where @id is an increment number. Then, all elements in that pool which are not associated with any lane will be associated with that lane. The reason for these two verifications is that, since the approach expects a text in the active voice, all elements necessarily need to be related to a resource (i.e., lane) which in turn must belong to a process (i.e., pool). Regarding *activity labels verification*, the activities are checked to see if they do not have a label. If unlabelled activities are identified, they are labeled as “Do activity with id @id” where @id is the id of the process element. Thus, even if a *process*, a *resource* or an *activity* is not defined in the *Process File*, the text will still be generated.

4.6.3 Modifications in DSynT - Message Generation Stage

In order to define the sentences that will compose the text, the first step was to make changes in the *DSynT - Message Generation stage*. In this stage some of the DSynTs created from the RPST *fragment* types (i.e., *trivial*, *bond*, *polygon*, *rigid*) were modified. These modifications will be presented taking into account each of the types of fragments.

Regarding the *trivial* fragment, no modifications were made. The reason for this is that DSynTs created from *trivial* are generated from information taken from the process without the need to create new sentences to supplement this information. The information that compose a *trivial* fragment are:

- **Action:** The sentence snippet that describes the action that is performed (i.e., verb).
- **Business Object:** The sentence snippet that describes the business object in which the action is performed.
- **Additional Information:** The sentence snippet that complements the action, with-

out being the verb that performs the action or the business object.

- **Role:** The sentence snippet that describes the role that performs the action.

As can be observed, the *Action*, the *Business object* and *Additional information* are extracted from the label of the activity itself. In addition, *Role* will be the participant that performs the process element being described. Thus, this information will be extracted from a lane in a BPMN 2.0 process model and inserted as the subject in the sentence. *Trivial* fragments may also be complemented with *prior information* for split and join gateways. In this case, no changes are required since this information is considered as a *bond* fragment and therefore will be defined by the *bond* fragment. As an example, the sentence represented in Figure 4.13 could have been generated from an activity “sends the form to the manager” present in a lane “employee”. Furthermore, this activity consists of the *Action* “sends”, the *Business object* “the form”, and the *Additional information* “to the manager”.

Figure 4.13: Example of sentence created from a trivial fragment.

$$\underbrace{\text{In any case}}_{\text{Prior information}}\text{, }\underbrace{\text{the employee}}_{\text{Role}}\ \underbrace{\text{sends}}_{\text{Action}}\ \underbrace{\text{the form}}_{\text{Business object}}\ \underbrace{\text{to the manager}}_{\text{Additional information}}\text{.}$$

Source: The authors

Bond fragment is used to construct sentences describing splits and joins. For this fragment, the sentence templates described in Table 4.2 were used. This table is composed of the type of process element being described (i.e., “Type”), if the gateway is of type split or join (i.e., “Scope”), the sentence template used in the previous work (i.e., “Leopold, Mendling and Polyvyanyy (2014) Approach”), and the sentence template suggested in the current work (i.e., “Current Approach”). If a join gateway is followed directly by another join gateway, the snippet “the process continues” will be used. As an example, to describe XOR-join followed by another XOR-join in Figure 3.3, it would be possible to create the sentence “In any case, the process continues.”.

In addition, the same sentence templates from the original approach was used to describe *skip* and *loop*. *Skip* seeks to represent different paths starting from a gateway where at least one of which is an empty path (i.e., no process elements from the split gateway to the respective join gateway). The only modification made to this case is that the original approach only allowed a single path beyond the empty path. In the new approach,

Table 4.2: Modifications in *bond* fragments.

Type	Scope	Leopold, Mendling and Polyvyanyy (2014) Approach	Current Approach
Exclusive Choice	split	One of the following branches is executed.	..., one of the *number* alternative procedures is executed.
	join	Once one of the alternative branches was executed ...	In any case, ...
Inclusive Choice	split	One or more of the following branches is executed	..., *number* alternative procedures may be executed.
	join	Once all desired branches were executed ...	Afterwards, ...
Parallelism	split	The process is split into *number* parallel branches.	..., *number* procedures are executed in an arbitrary order.
	join	Once all <number> branches were executed ...	After each case, ...
Skip	split	If required ...	If required, ...
	join	In any case ...	In any case, ...
Loop	split	-	-
	join	If required *role* repeats the latter steps and continues with... Once the loop is finished ...	If required *role* repeats the latter steps and continues with... Once the loop is finished ...

Source: The authors

multiple paths are allowed. For this, the sentence describing the split gateway has been modified to represent only the number of non-empty paths and, if it is an exclusive gateway or an inclusive gateway, add the term “if required” in the sentence. As an example, to describe the XOR-split in the process model of Figure 2.3 could be used the sentence “Then, *if required,* one of the *2* alternative procedure paths is executed.”. *Loop*, on the other hand, it was not modified since the only sentence template found to represent loops (i.e., St_{33}) had an ambiguity problem. In addition, the sentence templates presented in the Leopold, Mendling and Polyvyanyy (2014) approach help to evidence when the loop starts, runs, and terminates.

In the case of *polygon* fragment, since it represents only sequences of other fragments, the approach only separates the fragments that exist within a polygon fragment and processes them individually. Therefore, there are no sentences being created from *polygon* fragments and no modifications were made.

In case of *rigid* fragment, in order to understand how it is transformed into DSynTs, it is necessary to define *how the different paths of a rigid fragment are described* and *which are the sentences that compose a rigid fragment*. Regarding *how the different paths of a rigid fragment are described*, the paths are first transformed into a Petri net (MURATA, 1989). Thereafter, one path will be described from the beginning to the end of the fragment (i.e., main path) and then the other paths (i.e., alternative paths) will be described taking into account what has already been presented (i.e., $\mathbf{Case_{5-4}}$ present in Chapter 3).

Regarding *which are the sentences that compose a rigid fragment*, *rigid* fragments may be composed of other fragments (i.e., *trivial*, *bond*, and *polygon*) and some predefined sentence templates. When a fragment exists within the *rigid* fragment, it will be transformed into DSynT by its respective transformation approach. In this case, *rigid* can add extra information such as “may also” to indicate that it is describing a part of a path (e.g., “Then, the employee must fill out the report and send it to the financial manager. Next, the financial manager must sign confirming receipt. After completing the report, the clerk *may also* choose to send it to the financial sector secretariat.”). On the other hand, the original approach (LEOPOLD; MENDLING; POLYVYANYY, 2014) uses 3 different sentence templates to represent *rigid* fragments. These 3 sentence templates can be defined as follows:

- **Rigid introduction:** A sentence that initiates *rigid* making explicit that there is a region with many paths. For this case, the authors use the sentence template “Subsequently, the process contains a region which allows for different execution paths.”.

- **Main path introduction:** Introduction of the main path. For this case, the authors use the sentence template “One option from start to end is the following:”.

- **Alternative path introduction:** Introduction of other possible paths. For this case, the authors use the sentence template “However, the region also allows for a *number* deviations:”.

The original format to transform *rigid* fragments into DSynTs will be retained for long descriptions. However, for sentences containing few paths (i.e., defined as up to 5 paths) the Petri net will be traversed in order to describe each possible path from the beginning of the fragment to the respective end (i.e., $\mathbf{Case5-2}$)). In this case, extra information (i.e., “may also”) will not be required. In addition, the 3 sentence templates would be replaced by a single sentence template similar to the Exclusive Choice template in Table 4.2 (i.e., “..., one of the *number* alternative procedures is executed.”).

In addition to the modifications described above, some other customizations were made. To represent a *start event*, the sentence template “The process starts when ...”

(St_{58}) was used instead of “The process starts with ...” ($St_(57)$). In addition to being quite recurrent, this sentence template assures to use the same verb tense to describe all activities in the text (i.e., the sentence template St_{57} would require that only the first activity of the text be described in the infinitive). To describe the *start event* followed by a split gateway, the sentence template “When the process starts, ...” was used. Finally, “... the process ends.” was used to describe *end events* and could be complemented with terms like “Finally,”, “In any case,”, “Afterwards,”, and “After each case,”.

4.6.4 Modifications in Message Refinement Stage

For the *Message Refinement* stage, modifications were made in two techniques, being them: *Message Aggregation* and *Discourse Markers* (LEOPOLD, 2013). Tables 4.3 and 4.4 present the changes made in *message aggregation* and *discourse markers* respectively.

Regarding *Message Aggregation*, in order to avoid ambiguity the term “and then” was used instead of “and” to aggregate sentences of elements that appear sequentially. In addition, the possibility of aggregating elements belonging to different roles has been added (e.g., “The employee creates the report and then the manager signs.”).

Table 4.3: Modifications in message aggregation.

Message Aggregation	Leopold, Mendling and Polyvyanyy (2014) Approach	Current Approach
Default	A_c and A_c.	A_c and then A_c.
Different roles	-	$role_1$ A_c and then $role_2$ A_c.

Source: The authors

In relation to *Discourse Marker*, the discourse markers used to represent sequential elements have been modified to: “Next”, “Subsequently”, and “After that”. In addition, discourse markers to represent the different paths of a gateway were created.

For the representation of exclusive and inclusive paths where not all possible paths have a condition, the discourse marker “In the @ordinal procedure” was used. In this case, @ordinal will be replaced by the ordinal corresponding to the path currently being described (e.g., “In the *first* procedure, ...”). On the other hand, if all possible paths have a condition, the discourse marker “If @condition” was used. In this case, @condition will be replaced by the condition of the path currently being described (e.g., “If hardware problem, ...”).

For the representation of parallels paths, the first path has no discourse marker, but the others use discourse markers that convey the idea that different procedures occur at the same time. For this, the following discourse markers were used to indicate the idea of parallelism: “In the meantime” and “At the same time”. As pointed out in Section 3.4.3, these markers are considered to have ambiguity issues, but sentences that explicitly describes the beginning and end of parallelism can help to solve this problem.

Table 4.4: Modifications in discourse markers.

Discourse marker	Leopold, Mendling and Polyvyanyy (2014) Approach	Current Approach
Sequence	Then, ...	Next, ...
	Subsequently, ...	Subsequently, ...
	Afterwards, ...	After that, ...
XOR and OR Paths	-	In the @ordinal procedure, ...
	-	If “@condition”, ...
AND Paths	-	In the meantime, ...
	-	At the same time, ...

Source: The authors

4.6.5 Modifications in Realization Phase

As can be seen in Figure 4.12, the result of the *Text Writer Service* is no longer a natural language text to become a text with metadata. This metadata is information gather from the *Process File* and the generated text itself.

During the *DSynT - Message Generation stage*, the information gather of a process element will be associated with all DSynTs that represent that process element. This information can still be modified or even deleted according to the manipulation of DSynTs during the *Message Refinement stage*. Finally, in the *Realization phase*, each DSynT created by the *Sentence Planning phase* will be transformed into a sentence and, along with the information associated with it, will compose the *Text Metadata File*.

The exception to this generation process is the “startIndex” and “endIndex” attributes present in a *Text Metadata File*. These attributes will be created only during the *Realization phase* right after the DSynT is transformed into a sentence. This is required because, in order to define the indexes, it is necessary first to know how the sentence is.

4.6.6 Text Writer Output

Listing 4.7 presents the *Text Metadata File* in JSON format generated by *Text Writer Service* for the *Process File* represented in Listing 4.5. As can be seen, this document has two parts. The first part is a *list containing the processes and resources of each process* (lines 2 to 13) and the second part is a *list containing the sentences with metadata* (lines 14 to 149).

In the *list containing the processes and resources of each process*, it is possible to observe that there is a single process with id “id-0” (line 10) and name “Pool” (line 11). Within this process there is a resource with id “id-2” (line 6) and name “technician” (line 7) .

On the other hand, in the *list containing the sentences with metadata*, it is possible to observe that the text will consist of 7 sentences (lines 17, 41, 56, 71, 86, 110, and 134). As described in Section 4.2.3, each of these sentences will be accompanied by a boolean representing whether the sentence is *newSplitPath* and a *snippetList* . Each *snippetList*, in turn, will contain a *startIndex*, an *endIndex*, a *processElementId*, a *processElementType*, a *resourceId*, a *processId*, and a *level*. As an example, the sentence “*If \“a hardware problem\”, the technician replaces the part and then fills out the part replacement form.*” (Line 86) the value of *newSplitPath* is “true” (line 107) since it describes a path immediately after a gateway split. This sentence will still be composed of two snippets. The first snippet (lines 88 to 96) is represented in the sentence between the indices *startIndex* “25” and *endIndex* “57” (i.e., “the technician replaces the part”) and corresponds to the activity with id “id-5” being done by the resource with id “id-2” (i.e., “technician”), in the process with id “id-0” (i.e., “pool”), and having the value of *level* equals “1” (line 95). On the other hand, the second snippet (lines 97 to 105) is represented in the sentence between the indices *startIndex* “67” and *endIndex* “102” (i.e., “fills out the part replacement form”) and corresponds to the activity with id “6-id” being done by the resource with id “id-2” (i.e., “technician”), in the process with id “id-0” (i.e., “pool”), and having the value of *level* equals “1” (line 104).

Listing 4.7: Text metadata file example: Computer repair.

```
1 {
2   "processList": [
3     {
4       "resourceList": [
5         {
```

```
            "id": "id-2",
            "name": "technician"
          }
        ],
        "id": "id-0",
        "name": "Pool"
      }
    ],
    "text": {
      "sentenceList": [
        {
          "value": "The process starts when the technician performs an
              evaluation.",
          "snippetList": [
            {
              "startIndex": 0,
              "endIndex": 23,
              "processElementId": "id-11",
              "processElementType": "STARTEVENT",
              "resourceId": "id-2",
              "processId": "id-0",
              "level": 0
            },
            {
              "startIndex": 24,
              "endIndex": 61,
              "processElementId": "id-3",
              "processElementType": "ACTIVITY",
              "resourceId": "id-2",
              "processId": "id-0",
              "level": 0
            }
          ],
          "newSplitPath": false
        },
        {
          "value": "Next, one of the 3 alternative procedures is
              executed.",
          "snippetList": [
            {
              "startIndex": 6,
          "endIndex": 53,
          "processElementId": "id-9",
          "processElementType": "XORSPLIT",
          "resourceId": "id-2",
          "processId": "id-0",
          "level": 0
        }
      ],
      "newSplitPath": false
    },
    {
      "value": "If \"founds no problem\", the technician makes the no modification to the computer.",
      "snippetList": [
        {
          "startIndex": 24,
          "endIndex": 80,
          "processElementId": "id-7",
          "processElementType": "ACTIVITY",
          "resourceId": "id-2",
          "processId": "id-0",
          "level": 1
        }
      ],
      "newSplitPath": true
    },
    {
      "value": "If \"a software problem\", the technician formats the computer.",
      "snippetList": [
        {
          "startIndex": 25,
          "endIndex": 60,
          "processElementId": "id-4",
          "processElementType": "ACTIVITY",
          "resourceId": "id-2",
          "processId": "id-0",
          "level": 1
        }
      ],
      "newSplitPath": true
    },
    {
      "value": "If \"a hardware problem\", the technician replaces
          the part and then fills out the part replacement form.",
      "snippetList": [
        {
          "startIndex": 25,
          "endIndex": 57,
          "processElementId": "id-5",
          "processElementType": "ACTIVITY",
          "resourceId": "id-2",
          "processId": "id-0",
          "level": 1
        },
        {
          "startIndex": 67,
          "endIndex": 102,
          "processElementId": "id-6",
          "processElementType": "ACTIVITY",
          "resourceId": "id-2",
          "processId": "id-0",
          "level": 1
        }
      ],
      "newSplitPath": true
    },
    {
      "value": "In any case, the technician completes the repair
          form.",
      "snippetList": [
        {
          "startIndex": 0,
          "endIndex": 11,
          "processElementId": "id-10",
          "processElementType": "XORJOIN",
          "resourceId": "id-2",
          "processId": "id-0",
          "level": 0
        },
        {
          "startIndex": 13,
            "endIndex": 53,
            "processElementId": "id-8",
            "processElementType": "ACTIVITY",
            "resourceId": "id-2",
            "processId": "id-0",
            "level": 0
          }
        ],
        "newSplitPath": false
      },
      {
        "value": "Finally, the process ends.",
        "snippetList": [
          {
            "startIndex": 9,
            "endIndex": 25,
            "processElementId": "id-12",
            "processElementType": "ENDEVENT",
            "resourceId": "id-2",
            "processId": "id-0",
            "level": 0
          }
        ],
        "newSplitPath": false
      }
    ]
  }
}
```

4.7 Main Service

The *Main Service* is responsible to interact with other services in order to produce the business process-oriented text from the input provided by the user (i.e., process description or BPMN 2.0 process model). The service thus works by orchestrating the services deciding what and when each service should be called.

Initially, *Main Service* produces a web page and sends it to be processed by the user using a web browser. In this web page, the user can submit a request containing a *process description* or a *BPMN 2.0 process model*. Once the user provided a *process*

Figure 4.14: Example of rewritten process description: Computer repair.

The process starts when the technician performs an evaluation. Next, one of the 3 alternative procedures is executed. If "founds no problem", the technician makes the no modification to the computer. If "a software problem", the technician formats the computer. If "a hardware problem", the technician replaces the part and then fills out the part replacement form. In any case, the technician completes the repair form. Finally, the process ends.

Source: The authors

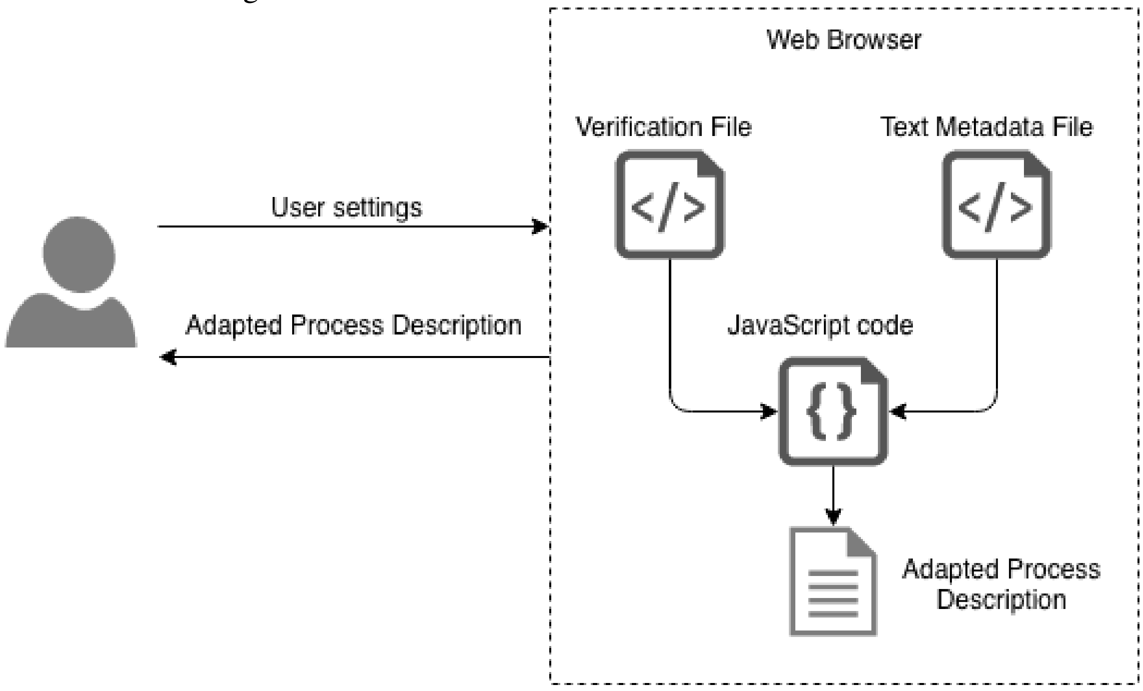
description as input, the *Main Service* must first interact with the *Text Reader Service* to identify the process elements and their relationships. Then, the *Main Service* interacts with the *Process Verification Service* and *Text Writer Service*. On the other hand, since the user provided a *BPMN 2.0 process model* as input, the service does not need to identify the process elements and their relationships, so it interacts directly with the *Process Verification Service* and *Text Writer Service*.

After receiving the verification made by the *Process Verification Service* and the text generated by the *Text Writer Service*, the *Main Service* produces the web page that will be sent to the user. This web page should contain the document produced by the *Process Verification Service* (i.e., *Process Verification File*), the document produced by the *Text Writer Service* (i.e., *Text Metadata File*), and the JavaScript code needed to process these documents and produce the text according to the user specifications. As an example the *Text Metadata File* in the Listing 4.7 can be processed by JavaScript code to produce the business process description shown in Figure 4.14. This business process description is produced by reading the values present in the *value* attributes of the *Text Metadata File*.

Figure 4.15 shows how occurs the interaction between the user and the web page provided by the *Main Service*. As can be seen, the user can interact with the web page by providing some settings. From the settings provided by the user, the JavaScript code will process the *Process Verification File* and the *Text Metadata File* to produce the process description adapted to the user settings.

In the current approach, the JavaScript code supports user settings that allow for three different features, being: *text marking*, *text structuring* and *process verification display*. As these features are provided by JavaScript code on the *Process Verification File* and the *Text Metadata File*, it is not necessary to communicate with the server or generate a new verification and text every time the user wants to perform a configuration on some of these features. Each of these features will be explored further below.

Figure 4.15: Interaction between user and web browser.



Source: The authors

4.7.1 Text Marking

The purpose of the *text marking* feature is to highlight in the text important parts of the process in order to facilitate their identification by the user. For this, the JavaScript code will process the *Text Metadata File* identifying the snippets that meet the settings imposed by the user. Whenever a snippet meets a setting, that sentence snippet will be highlighted in the sentence (i.e., from attribute *startIndex* to attribute *endIndex* in that *snippet*).

The current approach allows two different types of marking: *marking of the process element taking into account its type* and *marking of the process element taking into account the resource*.

The *marking of the process element taking into account its type* seeks to assign a specific color to sentence snippets representing process elements of a given type. Thus, if the user informs, for example, that he wants to identify all the *activities* of the business process-oriented text, the JavaScript code created will go through all the snippets of the sentences present in the *Text Metadata File* and highlight those snippets whose *processElementType* is “activity”.

The *marking of the process element taking into account the resource* seeks to attribute bold to the sentence snippets that represent the process elements that are associated with a particular resource. Thus, if the user informs, for example, that wants to identify all the process elements of the business process-oriented text that are related to the "manager" resource, the JavaScript code will go through all the snippets of the sentences present in the *Text Metadata File* and highlight those snippets whose *resource* is "manager".

4.7.2 Text Structuring

The purpose of the *text structuring* feature is to allow the user to interact with the text structuring it as he prefers. The current approach allows different types of text structuring, such as: *paragraph indentation*, *paragraph split*, *use of bullet points and numbering in different paths*.

Regarding *paragraph indentation*, this work gives the user the option of indenting new paragraphs. For this, the JavaScript code scrolls through the text and, every time it finds a new paragraph, decides based on the user's settings whether to indent it or not.

Regarding *paragraph split*, it is given to the user different options to choose when a new paragraph will be created. Among these options the user can choose if all text should be kept together, if there should be a new paragraph after a certain amount of words or sentences, or if a new paragraph should be created every time the *level* in the *Text Metadata File* vary (i.e., similar to Leopold, Mendling and Polyvyanyy (2012) approach). In addition, the user can choose whether to create a new paragraph when the JavaScript finds a new resource, a split gateway, or a new process path of a split gateway. Moreover, the user can choose if the different paths of a gateway should be kept together.

Regarding *use of bullet points and numbering in different paths*, the user can choose whether or not to use bullet points and numbering when describing different paths of a gateway. To do this, the JavaScript code scrolls through the *Text Metadata File* and assigns a marker whenever it encounters sentences that represents different paths of the same gateway. In this case, it is considered different paths of the same gateway the sentences that: precede and succeed sentences that describes the same split and join gateways, have equal value for the *level* property, and have the *newSplitPath* property with value "true".

4.7.3 Process Verification Display

The purpose of the *process verification display* feature is to display the verification messages identified by the *Process Verification Service* and stored in the *Process Verification File*. This feature can define whether the verification messages will be displayed or not in the text. In addition, verification messages can be filtered so that only verification messages related to a particular *source* or of a particular *message type* are displayed.

When filtered by a particular *source*, it may be chosen to display the *verification messages* described by a specific *source*. As presented in the Section 4.5, the current approach presents verification using two possible sources: *BPMN Verification* and *YAWL Verification*. Therefore, when the user select a certain set of *sources*, the JavaScript code created will scroll through the *Process Verification File* and produce a list containing only the *verification messages* whose *source* property matches one of the selected sources.

When filtered by a particular *message type*, it may be chosen to display the *verification messages* described by a specific type. As presented in the Section 4.5, the current approach has three possible message types: *Label*, *Structure*, and *Pragmatic*. Therefore, when the user select a certain set of *message types*, the JavaScript code created will scroll through the *Process Verification File* and produce a list containing only the *verification messages* whose *messageType* property matches one of the selected types.

After producing the list that meets the filter criteria related to the *source* and the *message type*, the *verification messages* can be presented to the user. In this work, the *verification messages* are displayed in two different formats. In the first format, a list is constructed that displays all the filtered *verification messages* grouped by their respective process elements (through property *processElementId*). In the second format, the *verification messages* are displayed in the text itself. For this, the *verification messages* of the *Process Verification File* are related to the *snippets* of the *Text Metadata File* through the property *processElementId* that both have.

4.8 Final Considerations

This chapter presented the service-oriented architecture for generation of business process-oriented text. In this context, this chapter presented the documents that compose the contracts (i.e., *data definitions files* and *service definitions files*), and the services that are constructed to meet the different functionalities of the approach.

The use of the architecture created is accompanied by different advantages. One of these advantages is the *compatibility* because the architecture allows the integration with services created in other programming languages. Files in the WADL and XSD format are not restricted to the JAVA language and can be manually or automatically transformed into code for other programming languages. In terms of *maintainability*, the architecture makes use of XSD files defined by the OMG to represent the BPMN 2.0 processes (i.e., *Process File Schema*). Thus, this approach uses for the definition of the *Process File* (i.e., representation of process elements and their relationships) the XSD files provided by the group responsible for defining the BPMN specification, which allows the creation of classes consistent with the current BPMN. Thereby, using these files helps keep the architecture up-to-date since for new versions of BPMN the approach just need to update the XSD files provided by OMG, automatically generate JAVA classes, and make the necessary modifications to the services.

For the construction of the services it was considered to use and adapt works existing in the literature. Adopting other approaches to the construction of services allows to make use of scientifically recognized works with tools that have a certain maturity and, in some cases, are widely used. However, the use of other approaches is also accompanied by the negative points present in them. One of these points is related to the 76.98% similarity of the *Text Reader Service*. Because the approach used is not able to achieve 100% similarity, incorrect process elements and relationships can be identified. Consequently, in case the business process-oriented text is generated from a natural language text, the text produced may contain inconsistencies since the output generated by the *Text Reader Service* (i.e., *Process File*) has inconsistencies. However, once the approach has been built as a SOA, new services can be created to contribute to or replace the current services. In that sense, all that is required is for the new approach to be transformed into a REST service that can meet the requirements of the contract.

In the following chapter will be presented the results obtained by the approach of generation of business process-oriented text.

5 TEST OF THE BUSINESS PROCESS-ORIENTED TEXT GENERATION ARCHITECTURE

This chapter presents the prototype built to make possible the test the architecture developed in Chapter 4. In addition, this chapter presents the results of the analysis that was performed to verify the similarity between processes and business process-oriented texts.

The remainder of this chapter is structured as follows. Section 5.1 presents the built prototype. Section 5.2 presents the results obtained from the application of the similarity verification technique based on graph edit distance. Finally, Section 5.3 presents the final considerations of this chapter.

5.1 Developed Prototype

In this section will be presented the implemented prototype. Firstly, we will present the technologies used to develop the prototype and provide a link where the prototype can be found. Then, the interface of the implemented prototype and its characteristics will be presented.

The prototype was developed using the Java programming language. Each of the 5 services defined in Chapter 4 was built as a REST service. Moreover, this prototype makes use of web technologies (i.e., HTML, CSS, and JavaScript) for the construction of its interface for user interaction. For the *service registry*, a REST-based service called Eureka[1] was used. Eureka was chosen because it is also developed in JAVA, in addition to being a simple tool and have been able to register services without the need for much configuration. Moreover, even the Eureka service registry is developed in JAVA, it allows services to be registered and searched through REST operations (i.e., which allows services in other programming languages to be used). The prototype developed is available in: <www.github.com/thanner/MainService>.

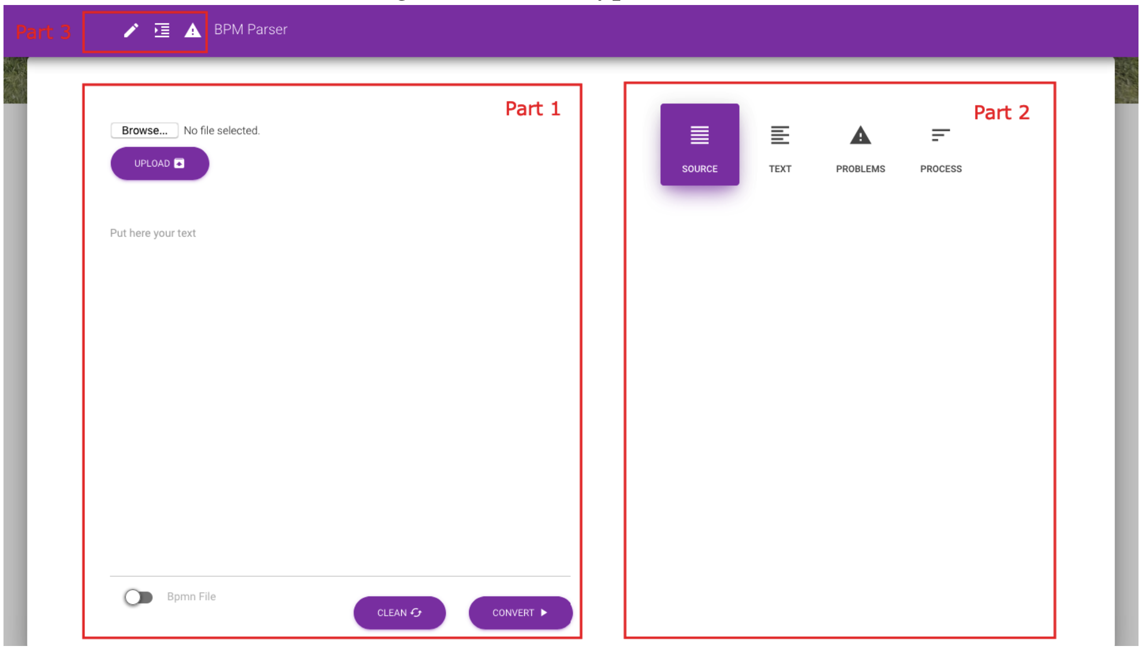
Regarding the interface of the implemented prototype, Figure 5.1 presents the overview of the prototype. As the figure shows, the prototype consists of three parts:

- **Part 1 - Input Data:** It is responsible for receiving an entry (i.e., natural language text or BPMN 2.0 file) and sending it to be processed by the *Main Service*.

[1]https://github.com/Netflix/eureka; last acessed 2019-02-02

- **Part 2 - Output Data:** It is responsible for displaying the outputs generated by the prototype.
- **Part 3 - Output Settings:** It is responsible for making the settings in the output data.

Figure 5.1: Prototype overview.

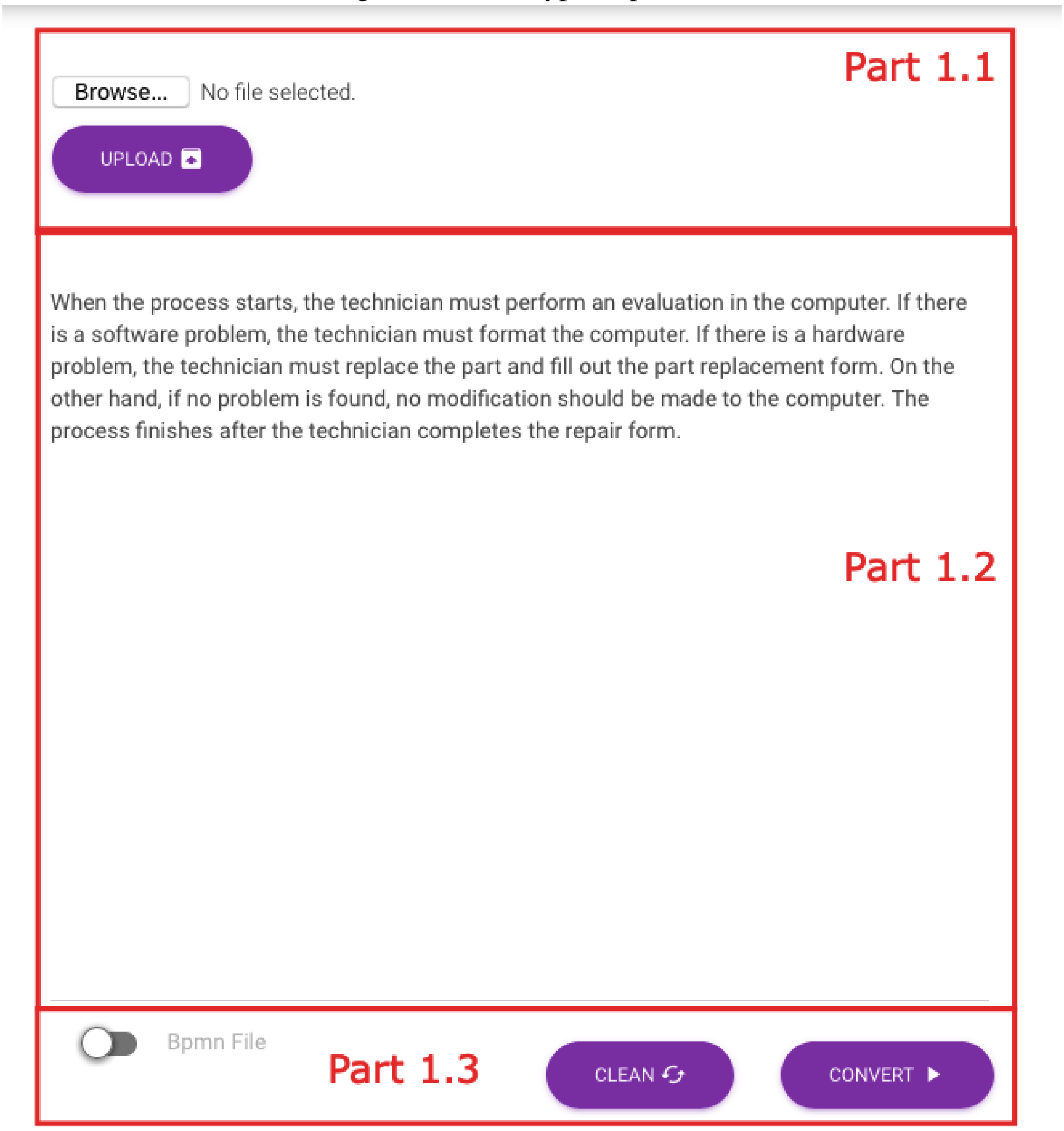


Source: The authors

Figure 5.2 presents Part 1 in more detail. As can be seen, Part 1 consists of three distinct areas. In the first area (i.e., Part 1.1) it is possible for a file to be loaded from the computer to the browser. In the second area (i.e., Part 1.2) it is possible to write the input data (i.e., natural language text or BPMN 2.0 file) or view the input data provided by the loaded file. Finally, in the third area (i.e., Part 1.3) it is possible to select whether the file sent is a natural language text or a BPMN file (i.e., BPMN File toggle button), clear the input data provided (i.e., CLEAN button), and send the input data to be processed by the *Main Service* (i.e., CONVERT button).

Part 2 consists of 4 tabs and the respective output of each tab. The first tab (i.e., SOURCE tab) displays the original text when the input provided is a natural language text. This original text can be marked (i.e., Section 4.7.1) and structured (i.e., Section 4.7.2) according to the user's output settings. If the input is a BPMN 2.0 file, the message "The original text does not exist." will be displayed. The second tab (i.e., TEXT tab) displays the business process-oriented text generated by the approach. Like the original text, this text can be marked and structured according to the user's output settings. In

Figure 5.2: Prototype: Input data.



When the process starts, the technician must perform an evaluation in the computer. If there is a software problem, the technician must format the computer. If there is a hardware problem, the technician must replace the part and fill out the part replacement form. On the other hand, if no problem is found, no modification should be made to the computer. The process finishes after the technician completes the repair form.

Source: The authors

addition, it may be chosen to display the identified process issues in this text. In this case, the sentence snippets that are related to process elements with verification messages will be underlined and the issues can be viewed through a tooltip when hovering the mouse over the snippet. The third tab (i.e., PROBLEMS tab) displays the verifications issues grouped by the process element ID (Figure 5.5). The fourth tab (i.e., PROCESS tab) displays the BPMN 2.0 file used for the verification and generation of business process-oriented text. This file may be the process produced by the *Text Reader Service* (i.e. if the input is natural language text) or the input itself (i.e., if the input is a BPMN 2.0 file).

Part 3 consists of three sidebars, from left to right: *Text Markers*, *Text structur-*

ing, and *Verifications*. The sidebar *Text Markers* is responsible for marking the process elements in the text. As can be seen in Figure 5.3, this sidebar is composed by toggle buttons for all possible process elements that can be marked. When the user activate a toogle button, its process element type will be identified and colored with the corresponding color in the original text (i.e., SOURCE tab) and in the business process-oriented text (i.e., PROCESS tab). The colors assigned to each process element can be easily modified in JavaScript code. For this prototype, each possible type of process element received a distinct color except splits and joins from the same gateway that, although they could be independently marked, received the same color. Since OMG does not define in its documentation the standard colors of the process elements, the colors used by the Bizagi process modeling tool to represent the process elements were used as the basis for the colors. Thus, the first four types of process elements (i.e., *activity*, *start event*, *intermediate event*, and *end event*) follow the colors that can be found for these process elements in a BPMN 2.0 process model made in the Bizagi tool. The other types of process elements (i.e., *XOR gateways*, *AND gateways*, *OR gateways*, *event-based gateway*), in turn, are the earlier colors with less brightness.

In addition, the sidebar *Text Markers* also has a drop-down list (i.e., RESOURCES) that lists all the resources found in the process. When a drop-down list element is selected, its resource will be identified and highlighted in bold in the original text (i.e., SOURCE tab) and in the business process-oriented text (i.e., PROCESS tab). By default, all toggle buttons and the drop-down in this sidebar start unchecked.

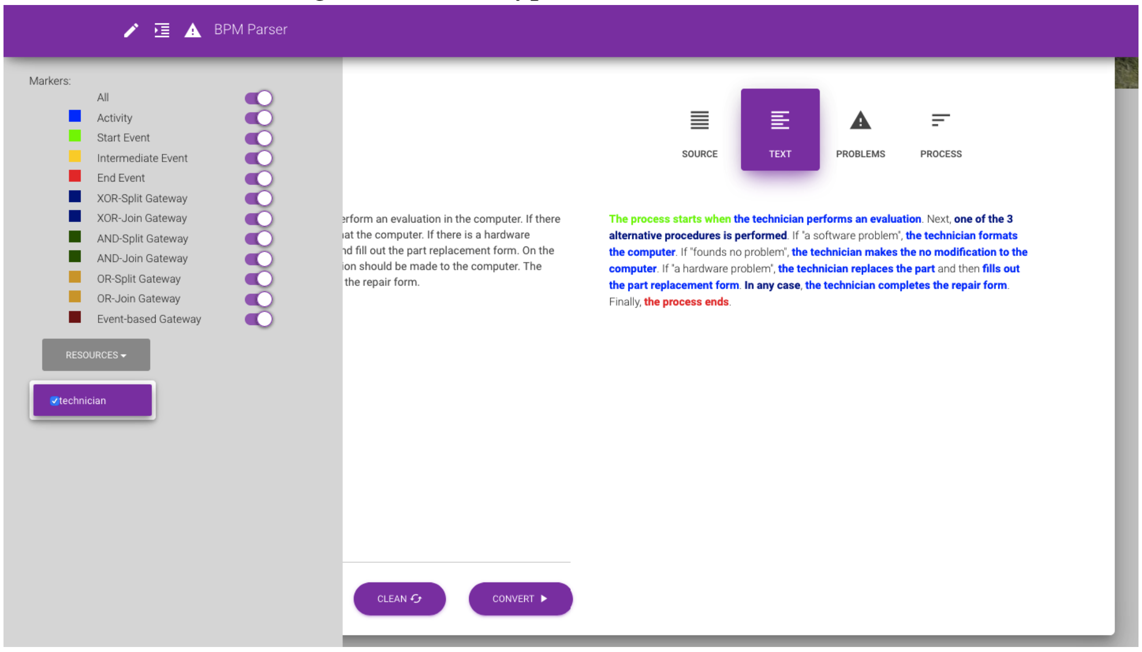
The sidebar *Text structuring* is responsible for defining the size of the paragraphs and the use of bullet points in the text. As can be seen in Figure 5.4, this sidebar is composed of six toggle buttons and two radio buttons. The toggle buttons, with their respective descriptions, can be seen as follows:

- **Split Resources:** Defines that a new paragraph must be created whenever there is a resource change between the previous sentence and the current sentence (i.e., who performs the actions). Since some sentences may involve more than one resource, the first resource described by the sentence will always be considered here.

- **Split Gateway:** Defines that a new paragraph must be created whenever a sentence describes a split gateway.

- **Split Procedure:** Defines that a new paragraph must be created whenever it describes another procedure (i.e., process path).

Figure 5.3: Prototype: Text Markers sidebar.



Markers:

	All	●
🟦	Activity	●
🟩	Start Event	●
🟨	Intermediate Event	●
🟥	End Event	●
🟦	XOR-Split Gateway	●
🟦	XOR-Join Gateway	●
🟩	AND-Split Gateway	●
🟩	AND-Join Gateway	●
🟧	OR-Split Gateway	●
🟧	OR-Join Gateway	●
🟥	Event-based Gateway	●

✓technician

erform an evaluation in the computer. If there
iat the computer. If there is a hardware
nd fill out the part replacement form. On the
ion should be made to the computer. The
the repair form.

The process starts when the technician performs an evaluation. Next, one of the 3 alternative procedures is performed. If "a software problem", the technician formats the computer. If "founds no problem", the technician makes the no modification to the computer. If "a hardware problem", the technician replaces the part and then fills out the part replacement form. In any case, the technician completes the repair form. Finally, the process ends.

Source: The authors

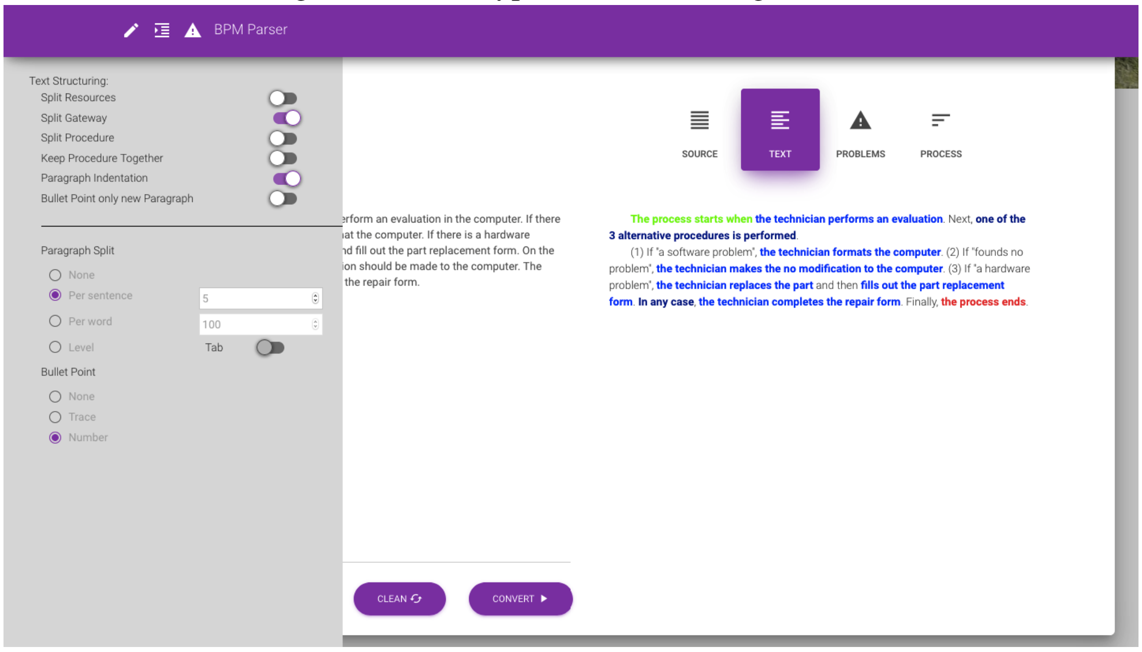
- **Keep Procedure Together:** Defines that process elements related to a procedure (i.e., path generated by a split gateway) are not separated into different paragraphs.

- **Paragraph Identation:** Defines whether paragraphs should start with indentation or not.

- **Bullet Point only new Paragraph:** Defines that, if bullet points are used, they appear only at the beginning of paragraphs.

The radio button *Paragraph Split* defines which criterion will be used to separate the paragraphs. The possible options are: *None* (i.e., no separations), *Per sentence* (i.e., by amount of sentences expected in a paragraph), *Per word* (i.e., by amount of words expected in a paragraph), and *Level* (i.e., similar to the Leopold, Mendling and Polyvyanyy (2014) approach presented in the Section 3.2.2). By choosing *Per sentence* or *Per word* options, the user can also set the number that corresponds to the desired amount of sentences or words. By choosing the *Level* option, the user can also define whether he want to use the same type of indentation of the Leopold, Mendling and Polyvyanyy (2014) approach (i.e., Tab toggle button). It should be noted that the *None*, *Per sentence*, and *Per word* criteria can be influenced by toggle buttons. Therefore, text configured to produce paragraphs with a size of 5 sentences can produce a smaller paragraph if the *Split Gateway* toggle button is active or a longer paragraph if the Keep Procedure Together toggle

Figure 5.4: Prototype: Text Structuring sidebar.



Text Structuring:
Split Resources
Split Gateway [✓]
Split Procedure
Keep Procedure Together
Paragraph Indentation [✓]
Bullet Point only new Paragraph

Paragraph Split
None
[✓ Per sentence] 5
Per word 100
Level Tab

Bullet Point
None
Trace
[✓ Number]

erform an evaluation in the computer. If there
iat the computer. If there is a hardware
nd fill out the part replacement form. On the
ion should be made to the computer. The
the repair form.

The process starts when the technician performs an evaluation. Next, one of the 3 alternative procedures is performed.
(1) If "a software problem", the technician formats the computer. (2) If "founds no problem", the technician makes the no modification to the computer. (3) If "a hardware problem", the technician replaces the part and then fills out the part replacement form. In any case, the technician completes the repair form. Finally, the process ends.

Source: The authors

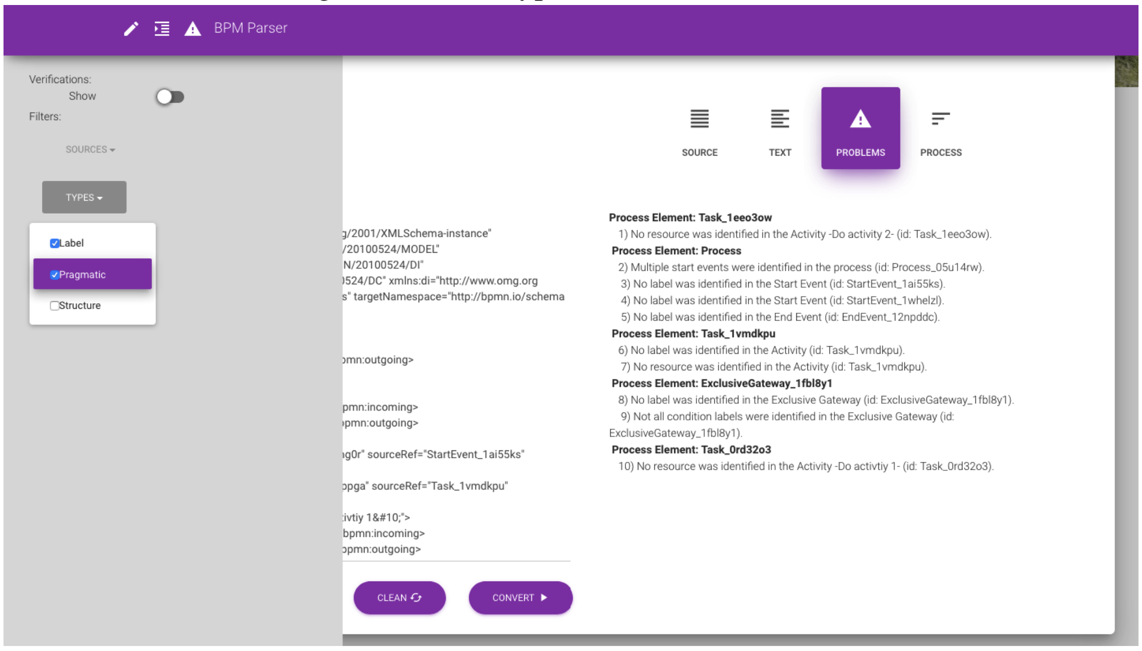
button is active. On the other hand, the *Level* criterion will not be influenced. In this case, by selecting the *Level* option the toggle buttons will be deselected and disabled.

The radio button *Bullet Point* defines how the different paths generated by a gateway will be marked. The possible options are: *None* (i.e., bullet points will not be used), *Trace* (i.e., a trace will be used as bullet point), and *Number* (i.e., a number will be used as bullet point). For the *Number* option, the first path after a gateway will receive the number “1” and the others will receive an increment of that number. In addition, in cases where there are nested gateways, numbers followed by dots will be used to represent the internal paths. As an example, considering the process model present in Figure 3.3, the path represented as “condition A” would receive the number “1”, the path represented as “condition B” would receive the number “2”, the path represented as “condition C” would receive the number “1.1”, and the path represented as “condition D” would receive the number “1.2”.

By default, all toggle buttons start unchecked. In addition, the *Paragraph Split* radio button starts by marking the option *Per sentence* and the number of sentences equal to 5. Moreover, the *Bullet Point* starts by marking the option *None*.

The sidebar *Verifications* is responsible for configuring which issues identified by the *Process Verification Service* will be displayed to the user. As can be seen in the Figure 5.5, this sidebar is composed of a toggle button and two drop-down lists. The toggle button (i.e., Show toggle button) defines whether the issues found should be also displayed

Figure 5.5: Prototype: Verifications sidebar.



```
g/2001/XMLSchema-instance"
/20100524/MODEL"
N/20100524/DI"
)524/DC" xmlns:di="http://www.omg.org
s" targetNamespace="http://bpmn.io/schema

omn:outgoing>

pmn:incoming>
pmn:outgoing>

ıg0r" sourceRef="StartEvent_1ai55ks"

ɔpga" sourceRef="Task_1vmdkpu"

ivtiy 1&#10;">
bpmn:incoming>
ɔpmn:outgoing>
```

Process Element: Task_1eeo3ow
1) No resource was identified in the Activity -Do activity 2- (id: Task_1eeo3ow).
Process Element: Process
2) Multiple start events were identified in the process (id: Process_05u14rw).
3) No label was identified in the Start Event (id: StartEvent_1ai55ks).
4) No label was identified in the Start Event (id: StartEvent_1whelzl).
5) No label was identified in the End Event (id: EndEvent_12npddc).
Process Element: Task_1vmdkpu
6) No label was identified in the Activity (id: Task_1vmdkpu).
7) No resource was identified in the Activity (id: Task_1vmdkpu).
Process Element: ExclusiveGateway_1fbl8y1
8) No label was identified in the Exclusive Gateway (id: ExclusiveGateway_1fbl8y1).
9) Not all condition labels were identified in the Exclusive Gateway (id: ExclusiveGateway_1fbl8y1).
Process Element: Task_0rd32o3
10) No resource was identified in the Activity -Do activtiy 1- (id: Task_0rd32o3).

Source: The authors

in the business process-oriented text (i.e., PROCESS tab). The first drop-down list (i.e., SOURCES drop-down list) should filter the issues so that only those whose source is the same as those marked in the drop-down list are displayed. Finally, the second drop-down list (i.e., TYPES drop-down list) should filter the issues so that only those whose type is the same as those marked in the drop-down list are displayed. If no options from a drop-down list are selected, the verification messages associated with this filter will not be filtered and then all verification messages will appear. By default, the toggle button and the drop-down lists start deselected.

5.2 Similarity Verification

In order to verify if the text generated by the approach is in accordance with its respective process, we performed a similarity verification between the *generated text* and the *original process*. This verification must take into account whether the process elements described in the *generated text* are equivalent to the process elements present in the *original process* and whether these process elements are equally related. Thus, the *generated text* will be considered similar to the *original process* when it has the same process elements as the *original process* and these process elements occur in the same order as the *original process*.

Considering that it is desired to identify the process elements and how they are related, the *original process* and the *generated text* were represented as process models. This format was chosen to perform the verification because the sequence flows present in a process model make it explicit how the process elements are related. To be represented as a process model, a given *generated text* must be transformed into a *generated process model*. In addition, to perform the verification, the *original process* must have a representation as the process model (defined here as *original process model*). Moreover, since the *generated text* produces a text from an existing process description, the *original process* also needs to have a representation as a textual description (defined here as *original text*).

Having the *generated text* and the *original process* represented as process models, we used an automatic verification technique based on similarity that assigns a value to how many different process models are equal. In the following sections, the verification procedure (Section 5.2.1) will be presented, as well as the results obtained from the verification (Section 5.2.2).

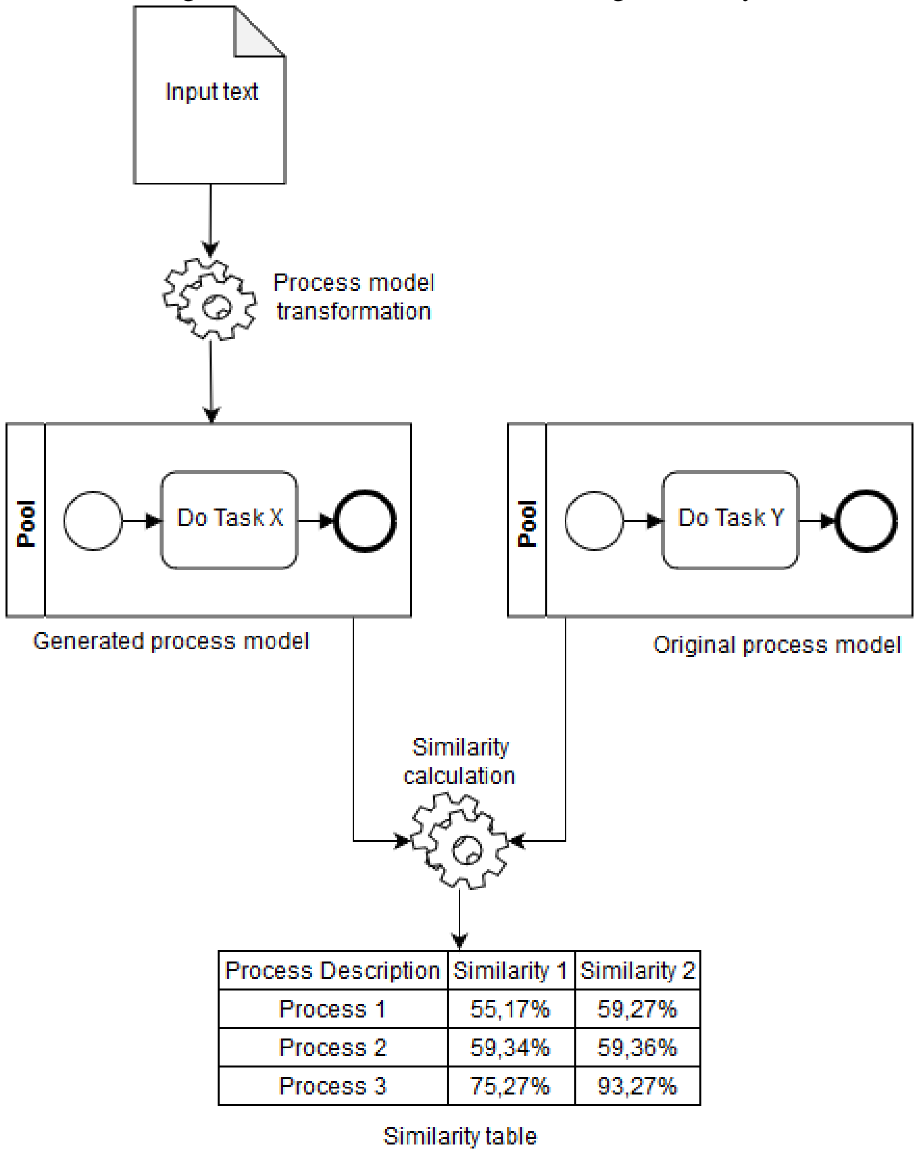
5.2.1 Verification Procedure

The procedure for calculating similarity is presented in Figure 5.6. In this figure, an *input text* is transformed into a BPMN 2.0 process model (i.e., *Process model transformation*). Then, the *original process model* is compared with the *generated process model* by a similarity technique (i.e., *Similarity calculation*). Finally, the results are stored in a table (i.e., *Similarity table*).

For the *Process model transformation*, the same technique present in *Text Reader Service* for identification of process elements in natural language texts was used (i.e., approach developed by Friedrich, Mendling and Puhlmann (2011)). The fact of using this transformation approach to convert the *input text* in BPMN 2.0 process model has some drawbacks, since it can lose information when doing the transformation. However, this approach was chosen because, in addition to the factors mentioned in Chapter 4, it allows to do this transformation automatically which contributes to the replication of the experiment. Another option would be to manually model the *input text* to then perform the similarity technique. However, the model produced, and consequently the result of similarity, would be influenced by the modeler's experience in modeling BPMN 2.0 processes, in understanding the proposed process descriptions, and in the English language.

Moreover, there are other approaches that allow us to calculate similarity without

Figure 5.6: Procedure for calculating similarity.



Process Description	Similarity 1	Similarity 2
Process 1	55,17%	59,27%
Process 2	59,34%	59,36%
Process 3	75,27%	93,27%

Source: The authors

having to turn the generated text into a process model (SÀNCHEZ-FERRERES; CARMONA; PADRÓ, 2017; AA; LEOPOLD; REIJERS, 2017). However, such approaches aimed at *aligning the natural language text with the process model* and not by defining a similarity between them. Using these approaches in order to verify the similarity between a natural language text and a process model could be undesirable because these approaches only compare the sentences of the text with the labels of the process model and do not take into account the structure of the process. Thus, in these approaches a sentence of a text and an activity of a process model can describe the same action (i.e., sharing of the same words) and therefore present a high similarity, even if the sentence describes that the action must occur at the beginning of the process and the activity describes that the action must occur at the end of the process. One option would be to adapt such approaches so that, after alignment, they define a value of general similarity between the text and the process. However, it would be necessary to create a new similarity technique. In addition, in none of these papers the proposed tool was found available.

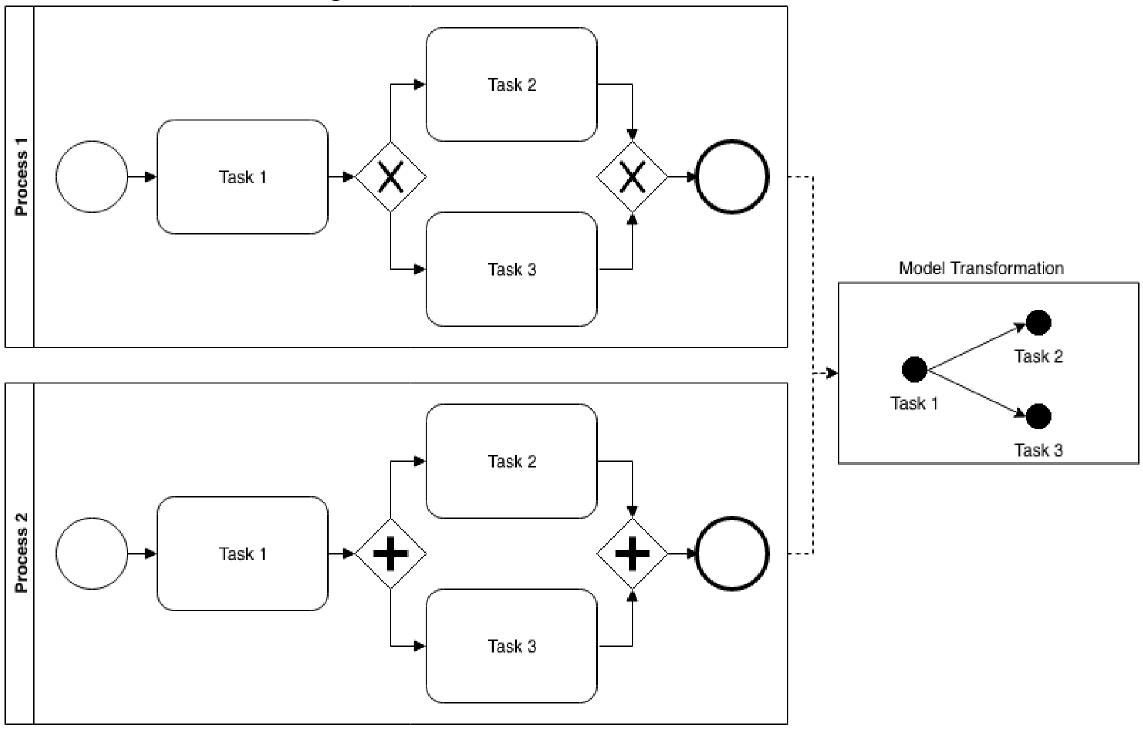
Regarding *similarity calculation* it is possible to find in the literature different techniques that allow to calculate the similarity between different process models (BECKER; LAUE, 2012; DUMAS; GARCÍA-BAÑUELOS; DIJKMAN, 2009). Similarity techniques generally compare two process models and assign a value to how much these process models are similar. In this sense, a technique can take into account different factors that it considers relevant (e.g., labels, amount of process elements, relationship between process elements), compare two process models and assign a value for when these process models are different (i.e., we will consider in this work low similarity as value 0), intermediate values for when they present resemblances, and another value for when they are equal (i.e., we will consider in this work high similarity as value 1). Among these techniques, the similarity was calculated taking into account the technique of *Graph edit distance similarity* (DIJKMAN; DUMAS; GARCÍA-BAÑUELOS, 2009).

The *graph edit distance* is defined as the minimal possible distance induced by a mapping between the two processes (DIJKMAN et al., 2011). The technique first makes a mapping between the process elements of two process models taking into account the labels of the elements. Then, the technique calculates how many operations (i.e., insert, delete, substitution of elements) are required for one process model to become equal to the other. For this, the technique takes the two process models and transforms them into graphs where the graph nodes (represented as N) are activities, events and gateways. In addition, the graph edges (represented as E) are the sequence flows. To perform the calculation, the approach considers three transformation operations: (1) *Node substitution* where a node from one graph is substituted for a node from the other graph; (2) *Node insertion/deletion* where a node is inserted into or deleted from a graph; (3) *Edge insertion/deletion* where an edge is inserted into or deleted from a graph.

The *graph edit distance similarity* is based on *graph edit distance* and allows comparing two process models assigning a value between 0 and 1 for how much these models are similar (i.e., the higher the number, the more similar the process models are). The similarity is given as $1 - grapheditdistance$. That is, the smaller the graph edit distance, the greater the *graph edit distance similarity*.

This technique was selected because it allows analyzing the labels of the process elements and how the process elements are related. In addition, the technique is described in the same programming language as the developed prototype (i.e., JAVA) and present itself as capable of verifying process models in different representations (including BPMN 2.0). Finally, the technique is publicly available in ProM process mining and analysis framework[2].

Figure 5.7: Process model transformation.



Source: The authors

However, this technique has some points that can be considered as disadvantages. One of these points is that it performs a transformation of the model into a graph where the gateways and events are disregarded. According to Dijkman, Dumas and García-Bañuelos (2009), these nodes can be disregarded during the similarity calculation process because they are one of those responsible for the combinatorial explosion when comparing process models, which affects this technique. However, Becker and Laue (2012) has already shown that this may lead to different process models being considered similar, which is undesirable in our verification. As an example, in the Figure 5.7, when the events and gateways are disregarded, both process models are converted to the same graph. However, there are different methods of transforming the process model into a graph (BECKER; LAUE, 2012). In the method we use, events and gateways that do not have a label receive their respective types as labels (e.g., “startevent”, “xorsplit”, “andjoin”, “endevent”). As

[2]http://prom.sourceforge.net; last acessed 2019-02-02

presented by Becker and Laue (2012) for the technique proposed by Grigori et al. (2010), inserting labels to these process elements allows assigning a slightly lower similarity for process models that are similar but have different types of gateways and events.

Another possible disadvantage of this technique is related to processing time. According to Becker and Laue (2012), techniques like *graph edit distance similarity* consume a long processing time because they analyze the complete structure of the process and this can be undesirable for identifying similar process models in a large repository. This does not present itself as a problem for our approach since the technique intends to compare only two process models for each process considered in this verification and does not conduct a search in a large repository with several process models. In addition, in order to find the mapping that induces the maximal similarity, we used the greedy algorithm presented in Dijkman, Dumas and García-Bañuelos (2009) since this algorithm presented a good mean average precision and a low execution time.

$$Similarity = 1.0 - \frac{wskipn \cdot fskipn + wskipe \cdot fskipe + wsubn \cdot fsubn}{wskipn + wskipe + wsubn} \tag{5.1}$$

According Dijkman, Dumas and García-Bañuelos (2009), the similarity can be given according to equation 5.1. For its equation, the following elements are presented:

- **wskipn:** weight assign to inserted or deleted nodes ($0 \leq wskipn \leq 1$).
- **fskipn:** fraction of inserted or deleted nodes. It is defined as the amount of inserted and deleted nodes ($|skipn|$) divided by the sum of the number of nodes of graph 1 ($|N_1|$) and graph 2 ($|N_2|$) (see Equation 5.2).
- **wskipe:** weight assign to inserted or deleted edges ($0 \leq wskipe \leq 1$).
- **fskipe:** fraction of inserted or deleted edges. It is defined as the amount of inserted and deleted edges ($|skipe|$) divided by the sum of the number of edges of graph 1 ($|E_1|$) and graph 2 ($|E_2|$) (see Equation 5.3).
- **wsubn:** weight assign to substituted nodes ($0 \leq wsubn \leq 1$).
- **fsubn:** average distance of substituted nodes. It is defined as the mapping (M) that assigns a similarity value between two nodes when checking the string edit distance between its labels ($Sim(n, m)$) divided by the amount of substituted nodes ($|subn|$). (see Equation 5.4).

$$fskipn = \frac{|skipn|}{|N_1| + |N_2|} \tag{5.2}$$

$$fskipe = \frac{|skipe|}{|E_1| + |E_2|} \tag{5.3}$$

$$fsubn = \frac{2.0 \cdot \sum_{(n,m) \in M} 1 - Sim(n, m)}{|subn|} \tag{5.4}$$

It is possible to customize the algorithm by assigning different values for $wskipn$, $wskipe$, and $wsubn$. Thus, different significance may be given for each operation (i.e., *Node insertion/deletion*, *Edge insertion/deletion*, and *Node substitution*). As well as Dijkman et al. (2011), the following values were assigned to the weights: $wskipn = 0.1$, $wskipe$ = 0.2, and $wsubn$ = 0.8.

As similarity techniques consider different factors to define when two models are similar (e.g., labels, structure, amount of nodes), different similarity techniques, or even different weights for the same similarity technique, can produce different results when comparing two process models. As an example, a similarity technique S_1 can consider that two process models have a similarity of 0.93 while a second technique S_2 can consider that the similarity between these same process models is 0.68. Thus, in order to compare the results obtained in this dissertation, the similarity will be presented for three different cases. Considering Figure 5.6, the procedure for calculating similarity will be the same for the three cases, changing in each case only the *Input text*. In the first case, the *Input text* will be the *original text*. In the second case, the *Input text* will be the *text generated from the original text*. In the third case, the *Input text* will be the *text generated from the original process model*. Finally, results obtained by calculating the similarity in each case were inserted into tables called *similarity tables*.

5.2.2 Results Obtained by the Similarity Technique

In order to perform the verification, the same process descriptions presented in Chapter 3 were used. Given the 81 process descriptions considered in this work (see Appendix), 15 have been disregarded since they do not have the original process model (Pd_{58} to Pd_{65}, and Pd_{75} to Pd_{81}). In addition, 4 were disregarded because the prototype used by us for the identification of process elements and their relationships was not able to read the text and produce a process model (Pd_1, Pd_{35}, Pd_{36}, and Pd_{70}). Thus, 62 business process descriptions were selected in order to perform this verification.

Taking into account the similarity technique employed and the different weights adopted, Table 5.1 presents the average similarity for the different data sources. In addition, Tables 5.2 and 5.3 present the results obtained for each business process description. These tables contain the id of the business process description (i.e., Pd_{ID}), the similarity between the original text and the original process model (i.e., *Original text*), the similarity between the text generated from the original text and the original process model (i.e., *Generated from text*), and the similarity between the text generated from the original process model and the original process model (i.e., *Generated from model*).

Table 5.1: Similarity summary.

Source	**Original Text**	**Generated from Text**	**Generated from Model**
Friedrich (2010)	73.02%	71.15%	71.88%
Dumas et al. (2013)	59.59%	55.13%	64.44%
Total	69.16%	66.50%	69.73%

Source: The authors

As can be seen, was obtained a greater similarity in business process descriptions extracted from Friedrich (2010) dissertation than from business process descriptions extracted from Dumas et al. (2013) book. In analyzing some of the worst results from business process descriptions extracted from Dumas et al. (2013) it is possible to observe that this results is connected to the approach of identification of process elements since it influences the similarity of *original text* and text *generated from text*, but do not have an impact on the text *generated from model* (i.e., Pd_{48}, Pd_{66}, Pd_{72}). Since the business process descriptions present in Friedrich (2010) were the same as those used by the approach for process element identification, it is expected that common structures present in these business process descriptions (e.g., some stop words used to identify a parallelism or an exclusivity in a process description) where assimilated by the approach. Another factor that reinforces this reasoning is that, according to Table 5.1, for process descriptions whose source is “Dumas et al. (2013)”, texts *generated from models* presented similarity considerably superior to the *original texts* and texts *generated from text* (something that is not observed for process descriptions whose source is “Friedrich (2010)”). Thus, this can be considered as an opportunity for improvement in the process element identification approach (i.e., used in this work in the *Text Reader Service*) since texts extracted from Dumas et al. (2013) may reveal new ways of describing business processes and, conse-

Table 5.2: Similarity - Part 1.

$\mathrm{Pd}_{\mathrm{ID}}$	**Original Text**	**Generated from Text**	**Generated from Model**
Pd_2	75.76%	74.64%	74.21%
Pd_3	74.71%	74.19%	74.66%
Pd_4	74.10%	73.39%	74.95%
Pd_5	73.59%	73.74%	73.89%
Pd_6	74.45%	74.04%	73.71%
Pd_7	73.55%	73.55%	73.86%
Pd_8	76.36%	76.22%	76.36%
Pd_9	76.03%	76.03%	74.75%
Pd_{10}	76.56%	74.75%	75.00%
Pd_{11}	73.22%	73.25%	73.52%
Pd_{12}	76.36%	75.32%	74.18%
Pd_{13}	75.00%	75.00%	75.00%
Pd_{14}	75.42%	74.18%	74.38%
Pd_{15}	72.92%	72.90%	73.08%
Pd_{16}	73.55%	73.55%	74.03%
Pd_{17}	74.64%	73.68%	74.75%
Pd_{18}	73.54%	73.55%	73.35%
Pd_{19}	73.59%	73.55%	75.00%
Pd_{20}	73.78%	73.50%	73.90%
Pd_{21}	75.76%	75.76%	75.52%
Pd_{22}	73.28%	73.25%	73.68%
Pd_{23}	78.11%	73.52%	74.80%
Pd_{24}	55.50%	37.32%	55.41%
Pd_{25}	63.55%	40.41%	46.75%
Pd_{26}	55.24%	55.24%	46.46%
Pd_{27}	46.58%	33.08%	47.61%
Pd_{28}	74.75%	74.64%	73.57%
Pd_{29}	74.29%	74.33%	74.03%
Pd_{30}	73.45%	73.48%	73.28%
Pd_{31}	73.38%	73.40%	73.71%
Average	73.02%	71.15%	71.88%

Source: The authors

quently, the existing approach can be refined to become even more effective in identifying process elements and their relationships.

In addition, it is possible to observe that the similarity of the texts *generated from text* presented results slightly inferior to *original texts*. This was expected since, as well as the *original text*, the text *generated from text* also depends on the approach of identifying process elements in a text. However, although slightly inferior, the text *generated from text* (just as texts *generated from model*) has different advantages, such as: use of recurring sentence templates, less potential to present ambiguity issues, and allow the marking of

Table 5.3: Similarity - Part 2.

$\mathrm{Pd}_{\mathrm{ID}}$	**Original Text**	**Generated from Text**	**Generated from Model**
Pd_{32}	73.66%	73.28%	73.86%
Pd_{33}	73.68%	73.62%	73.29%
Pd_{34}	76.36%	74.64%	74.38%
Pd_{37}	76.95%	73.48%	73.41%
Pd_{38}	75.15%	75.15%	75.52%
Pd_{39}	76.36%	74.64%	74.38%
Pd_{40}	74.68%	74.07%	73.94%
Pd_{41}	74.75%	74.13%	74.03%
Pd_{42}	76.36%	74.03%	73.98%
Pd_{43}	74.69%	73.77%	73.64%
Pd_{44}	74.69%	73.74%	73.68%
Pd_{45}	76.09%	74.24%	74.07%
Pd_{46}	74.75%	74.87%	74.38%
Pd_{47}	75.91%	73.37%	73.32%
Pd_{48}	28.06%	11.22%	74.31%
Pd_{49}	74.31%	74.24%	74.31%
Pd_{40}	50.66%	49.70%	74.87%
Pd_{51}	73.38%	73.43%	74.68%
Pd_{52}	55.41%	55.41%	33.72%
Pd_{53}	73.61%	73.21%	73.64%
Pd_{54}	55.68%	55.68%	52.01%
Pd_{55}	73.68%	73.68%	58.15%
Pd_{56}	73.64%	73.68%	50.67%
Pd_{57}	73.84%	73.82%	73.68%
Pd_{66}	12.04%	11.90%	57.23%
Pd_{67}	73.17%	27.73%	48.93%
Pd_{68}	73.55%	73.48%	73.38%
Pd_{69}	73.22%	73.23%	73.17%
Pd_{71}	42.65%	26.17%	73.48%
Pd_{72}	18.94%	18.88%	74.31%
Pd_{73}	73.10%	73.11%	73.34%
Pd_{74}	73.68%	73.74%	46.05%
Average	59.59%	55.13%	64.44%

Source: The authors

process elements in sentences snippets.

Moreover, it is important to note that texts *generated from model* presented a similarity slightly superior to the *original texts*, which suggests the ability to produce more structured texts by improving the process element identification approach. Thus, one point that may be interesting would be the process analyst already describing the process following the structure defined in Chapter 3. In any case, it is important to remember that *the purpose of this work is not to produce the process model. The purpose is to produce* *a description that can be used as a resource for communication between process analyst and the domain expert as a step prior to process modeling.*

In addition to improving the *Text Reader Service*, other steps can be taken toward obtaining a greater similarity value for the approach presented in this dissertation. A possible step would be to increase the scope of process elements considered in this work. Since some process elements are not representable by the current approach, some process elements may not have been properly identified in a business process description by the *Text Reader Service* or may not have been properly transformed into a sentence by the *Text Writer Service*. Another step would be to identify whether the original business process description and the original process model are at the same level of abstraction. If one presents itself as more detailed than the other, comparing similarity results may be impaired.

5.3 Final Considerations

This chapter presents the prototype constructed from the approach proposed by this work. In addition, this chapter presents the verification technique based on similarity that verifies if the generated text is structured according to the business process.

As it was presented, the prototype interface consists of a single web page with different tabs that allow the manipulation of the files generated by the services (i.e., *Process Verification File*, and *Text Metadata File*). This prototype can be complemented in order to provide new user interactions, such as marking other process elements in the business process description, adding other possibilities of text structuring, or even presenting to the user the respective BPMN 2.0 process model.

Regarding the verification technique based on similarity, a method was presented that allows to automatically define how texts produced by the approach are structurally similar to their respective business processes (i.e., represented through the original process models). This validation was made in business process descriptions obtained from two different sources and presented good results, but also demonstrated some possible points of improvement. In addition, we have presented factors that may have influenced the similarity results and possible steps that can be taken to increase similarity and, consequently, generate more structured business process descriptions.

In the following chapter will be presented the conclusions of this work.

6 CONCLUSION

This work brings as a contribution an approach that allows the generation of business process-oriented text from natural language text. In this work, a business process-oriented text is a text that is structured, able to maintain the maximum information related to the business process, and able to check the quality of the process in relation to the BPMN 2.0 and in relation to soundness. For this, empirical analyses were performed to determine how a text should be constructed (Chapter 3). In addition, an SOA-based architecture was developed in which the steps required to generate the text are addressed as individual services (Chapter 4). In order to make possible the test of the developed architecture, a prototype was built. In addition, the process description produced by the approach was compared to the original process model through a similarity technique based on graph-edit distance (Chapter 5).

The following specific objectives have been achieved:

- *Define how a business process-oriented text should be*: For that, in Chapter 3 analyses were performed in order to define how the text should be structured and which sentences should be used to compose a business process-oriented text.
- *Define an architecture that allows the generation of business process-oriented text*: For that, the Chapter 4 presents a SOA architecture composed of 3 contracts and five services that enables the generation of business process-oriented texts.
- *Check the similarity of the business process-oriented text to the original process*: An analysis was performed using a graph edit distance similarity technique to determine if the text produced by the approach is in accordance with the original business process model.

In addition, the main contributions of this work are:

- In order to define how a process description should be, empirical analyses of the literature and existing business descriptions have been developed. This analysis sought to answer five different questions related to business processes descriptions. In addition, we presented 101 sentence templates that were identified, quantified, categorized, and verified in terms of ambiguity issues. Both the analyses, as well as the design defined in this work, can be useful for approaches that generate business process descriptions or approaches that perform process discovery from business process descriptions.

- Creation of an architecture for generation of business process-oriented text. This architecture is based on SOA and consists of 5 services that allow: the storage and discovery of services (i.e., *Service Registry*), the identification of process elements and their respective relationships in a business process description (i.e., *Text Reader Service*), the verification of the process according to BPMN 2.0 and soundness (i.e., *Process Verification Service*), the generation of a business process description following the text design defined in Chapter 3 (i.e., *Text Writer Service*), and the interaction with the user and with the other services (i.e., *Main Service*).
- In order to make possible the test of the developed architecture in Chapter 4, a prototype that allows the generation of business process-oriented texts was implemented. Moreover, this prototype allows the marking of process elements in a process description and structuring of text.

Despite the contributions presented above, this work presents some possible limitations. The first is related to the limited set of process elements considered by the work. Even using the most recurring process elements in BPMN 2.0, the notation has several process elements and their presence in a process description or a process model may not be properly considered by the current approach. In this sense, extending the approach to more process elements can contribute to a more complete business process-oriented text. Another limitation could be the fact that the analyses for the design of text performed in Chapter 3 are carried out manually. Even considering the problems pointed out in an automatic analysis, once the procedure for identifying and classifying sentence templates has already been defined in this work, an automatic analysis could collect information from a larger set of process descriptions and find new sentence templates. Thus, the development of an automatic analysis presents opportunities for future works. Finally, once a prototype has been developed, other validations beyond the similarity technique could be made. One option would be a scenario in which users could interact with the prototype or with the business process-oriented text generated by it.

Nevertheless, the work showed promising results regarding the generation of business process descriptions. Since we used recurrent and filtered sentence templates, we tried to produce process descriptions in a way that is closer to a pattern and with less ambiguity issues. In addition, given the architecture developed, the current services can be enhanced and new services can be made available to complement the approach. Finally, as stated earlier, the purpose of this work is not to produce a process model, but rather to supplement the information of a process or be used by process analyst and domain experts as a document during the *process discovery phase*. Therefore, the goal is not that this document should transform or replace a process model. *The goal is for this document to be confronted with the domain expert in order to better understand how a process occurs in an organization.*

The following possibilities has been identified as future works:

- Regarding process description design analysis in Chapter 3, the approach could be augmented with new samples of process descriptions (i.e., currently 64 process descriptions).

- Regarding the sentence template analysis, NLP techniques can be used to automatically extract sentence templates from business process descriptions.

- The analysis of sentence templates developed in Chapter 3 could be updated to be able to represent a greater amount of process elements and new cases of ambiguity issue.

- The contracts defined in Chapter 4 could be updated by identifying new ways of representing and manipulating business process descriptions.

- The services developed in Chapter 4 could also be updated to be able to represent a greater amount of process elements.

- With respect to the *Text Reader Service*, functionalities could be developed with the aim of improving the similarity of texts based on the points of possible improvements discussed in Section 4.4.3.

- With regard to *Process Verification Service*, it could increase the possible verifications made by the approach. For this, new algorithms could be created to verify process descriptions or to use other techniques such as BPMN ontologies (ROSPOCHER; GHIDINI; SERAFINI, 2014; NATSCHLÄGER, 2011) to verify the process described.

- Regarding the prototype, new forms of user interaction with the business process-oriented text could be created. One option would be to allow the user to interact with the input provided without having to perform the processing again. One possibility for this would be to consider the id of the process element. Thus, by modifying the label of an activity, it might not be necessary to do the entire procedure to identify process elements, checking the process, and generating the text for the whole business process again.

6.1 Publications

This section presents the scientific productions related to this dissertation. In all, three papers were produced and will be described in chronological order. For each paper will be presented the name of the paper, the authors, the Conference/Journal, the year of publication, and the Qualis[1] at the time of submission.

The first two papers were about topics related to the application of NLP in BPM, in which this author participated as writer or co-writer of the paper. The third paper is related to empirical analysis of sentence templates and ambiguity issues for business process descriptions (i.e., Section 3.3).

- **Recognition of Business Process Elements in Natural Language Texts.**
  Authors: Renato César Borges Ferreira, Thanner Soares Silva, Diego Toralles Avila, Lucinéia Heloisa Thom, and Marcelo Fantinato.
  Conference/Journal: Springer Book.
  Year of publication: 2018.
  Qualis: N/A.

- **Natural Language Processing in Business Process Identification and Modeling: A Systematic Literature Review.**
  Authors: Ana Cláudia de Almeida Bordignon, Lucinéia Heloisa Thom, Thanner Soares Silva, Vinicius Stein Dani, Marcelo Fantinato, and Renato Cesar Borges Ferreira.
  Conference/Journal: XIV Brazilian Symposium on Information Systems - SBSI 2018.
  Year of publication: 2018.
  Qualis: B2.

- **Empirical Analysis of Sentence Templates and Ambiguity Issues for Business Process Descriptions.**
  Authors: Thanner Soares Silva, Lucinéia Heloisa Thom, Aline Weber, José Palazzo

[1]Brazilian system of quality evaluation of the intellectual production of graduate programs.

Moreira de Oliveira, and Marcelo Fantinato.

Conference/Journal: 26th International Conference on Cooperative Information Systems - CoopIS 2018.

Year of publication: 2018.

Qualis: A2.

7 APPENDIX: BUSINESS PROCESS DESCRIPTIONS

Tables 7.1 to 7.3 present the business process descriptions used in this work. Each table has a business process description identifier (i.e., Pd_{ID}), the name of the business process description, and whether the business process description is accompanied by some process model. Business process descriptions are available at: <https://github.com/thanner/ProcessDescriptions>.

Table 7.1: Business process descriptions 1.

$\mathbf{Pd_{ID}}$	**Business Process Description Name**	**Contains Model**
Business process descriptions by Friedrich (2010)		
Pd_1	Bicycle manufacturing	yes
Pd_2	Computer repair	yes
Pd_3	Hotel Service	yes
Pd_4	Underwriters	yes
Pd_5	SLA Violation	yes
Pd_6	Supplier Switch	yes
Pd_7	MC Finalise SCT Warrant Posession	yes
Pd_8	Conduct Directions Hearing	yes
Pd_9	Repetition Cycles	yes
Pd_{10}	Event-based Gateways	yes
Pd_{11}	P&E - Lodge Originating Document by Post	yes
Pd_{12}	Claims Notification	yes
Pd_{13}	Claims Creation	yes
Pd_{14}	Claims Handling Process	yes
Pd_{15}	Intaker Work	yes
Pd_{16}	Active VOS Tutorial	yes
Pd_{17}	BizAgi Tutorial 1	yes
Pd_{18}	BizAgi Tutorial 2	yes
Pd_{19}	Oracle Tutorial	yes
Pd_{20}	ACME	yes
Pd_{21}	Inubit AG Tutorial	yes
Pd_{22}	Powerlicht	yes
Pd_{23}	Turbopixel	yes
Pd_{24}	Calling Leads	yes
Pd_{25}	HR Process - Simple	yes
Pd_{26}	HR Process - HR Department	yes
Pd_{27}	HR Process - Functional Department	yes
Pd_{28}	Exercise 1	yes
Pd_{29}	Exercise 2	yes
Pd_{30}	Exercise 3a	yes
Pd_{31}	Exercise 3b	yes

Source: The authors

Table 7.2: Business process descriptions 2.

$\mathrm{Pd}_{\mathrm{ID}}$	**Business Process Description Name**	**Contains Model**
Pd_{32}	Exercise 4	yes
Pd_{33}	Exercise 5	yes
Pd_{34}	Process B2	yes
Pd_{35}	Process B3	yes
Pd_{36}	Process B4	yes
Pd_{37}	Process B5.1	yes
Pd_{38}	Process B5.2	yes
Pd_{39}	Process B6	yes
Pd_{40}	Process B7	yes
Pd_{41}	Process B8	yes
Pd_{42}	Process C1	yes
Pd_{43}	Process C2	yes
Pd_{44}	Process C3	yes
Pd_{45}	Process D1	yes
Pd_{46}	Process D2	yes
Pd_{47}	Process D3	yes
Business process descriptions by Dumas et al. (2013)		
Pd_{48}	Example 3.2	yes
Pd_{49}	Exercise 3.1	yes
Pd_{50}	Example 3.3	yes
Pd_{51}	Exercise 3.2	yes
Pd_{52}	Example 3.5	yes
Pd_{53}	Example 3.6	yes
Pd_{54}	Exercise 3.3	yes
Pd_{55}	Example 3.7	yes
Pd_{56}	Exercise 3.4	yes
Pd_{57}	Example 3.8	yes
Pd_{58}	Exercise 3.10	no
Pd_{59}	Exercise 3.11	no
Pd_{60}	Exercise 3.12	no
Pd_{61}	Exercise 3.13	no
Pd_{62}	Exercise 3.18	no
Pd_{63}	Exercise 3.19	no
Pd_{64}	Exercise 3.20	no
Pd_{65}	Example 4.1	no
Pd_{66}	Exercise 4.4	yes
Pd_{67}	Exercise 4.5	yes
Pd_{68}	Example 4.3	yes
Pd_{69}	Exercise 4.7	yes

Source: The authors

Table 7.3: Business process descriptions 3.

$\mathbf{Pd_{ID}}$	**Business Process Description Name**	**Contains Model**
Pd_{70}	Exercise 4.8	yes
Pd_{71}	Exercise 4.11	yes
Pd_{72}	Exercise 4.12	yes
Pd_{73}	Exercise 4.14	yes
Pd_{74}	Exercise 4.16	yes
Pd_{75}	Exercise 4.18	no
Pd_{76}	Exercise 4.22	no
Pd_{77}	Exercise 4.28	no
Pd_{78}	Exercise 4.30	no
Pd_{79}	Exercise 4.31	no
Pd_{80}	Exercise 4.33	no
Pd_{81}	Exercise 4.34	no

Source: The authors

REFERENCES

AA, H. van der; LEOPOLD, H.; REIJERS, H. A. Detecting inconsistencies between process models and textual descriptions. In: SPRINGER. *International Conference on Business Process Management*. [S.l.], 2015. p. 90–105.

AA, H. van der; LEOPOLD, H.; REIJERS, H. A. Dealing with behavioral ambiguity in textual process descriptions. In: SPRINGER. *International Conference on Business Process Management*. [S.l.], 2016. p. 271–288.

AA, H. van der; LEOPOLD, H.; REIJERS, H. A. Comparing textual descriptions to process models–the automatic detection of inconsistencies. *Information Systems*, Elsevier, v. 64, p. 447–460, 2017.

AA, H. van der; LEOPOLD, H.; REIJERS, H. A. Checking process compliance against natural language specifications using behavioral spaces. *Information Systems*, Elsevier, 2018.

AALST, W. M. V. D. Business process management: a comprehensive survey. **ISRN Software Engineering**, Hindawi Publishing Corporation, v. 2013, 2013.

AALST, W. M. V. D.; HOFSTEDE, A. H. T. Yawl: yet another workflow language. **Information systems**, Elsevier, v. 30, n. 4, p. 245–275, 2005.

AALST, W. M. van D. et al. Workflow patterns. **Distributed and parallel databases**, Springer, v. 14, n. 1, p. 5–51, 2003.

AALST, W. M. van D.; PESIC, M.; SCHONENBERG, H. Declarative workflows: Balancing between flexibility and support. **Computer Science-Research and Development**, Springer, v. 23, n. 2, p. 99–113, 2009.

AALST, W. M. Van der. The application of petri nets to workflow management. **Journal of circuits, systems, and computers**, World Scientific, v. 8, n. 01, p. 21–66, 1998.

AALST, W. M. Van der. Formalization and verification of event-driven process chains. **Information and Software technology**, Elsevier, v. 41, n. 10, p. 639–650, 1999.

AALST, W. M. van der. Business process management: a personal view. **Business Process Management Journal**, Emerald Group Publishing Limited, v. 10, n. 2, 2004.

AALST, W. Van der; WEIJTERS, T.; MARUSTER, L. Workflow mining: Discovering process models from event logs. **IEEE Transactions on Knowledge & Data Engineering**, IEEE, n. 9, p. 1128–1142, 2004.

ADAMS, M.; HOFSTEDE, A. ter. Yawl user manual, version 4.1. **User manual, The YAWL Foundation**, 2016.

AKBAR, S.; BAJWA, I. S.; MALIK, S. Scope resolution of logical connectives in nl constraints. In: IEEE. **Eighth International Conference on Digital Information Management**. [S.l.], 2013. p. 217–222.

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.

BAKER, C. F.; FILLMORE, C. J.; LOWE, J. B. The berkeley framenet project. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 17th international conference on Computational linguistics-Volume 1**. [S.l.], 1998. p. 86–90.

BALANI, N.; HATHI, R. **Apache Cxf web service development: Develop and deploy SOAP and RESTful web services**. [S.l.]: Packt Publishing Ltd, 2009.

BEARD, R.; VOLPE, M. Lexeme-morpheme base morphology. In: **Handbook of word-formation**. [S.l.]: Springer, 2005. p. 189–205.

BECKER, M.; LAUE, R. A comparative survey of business process similarity measures. **Computers in Industry**, Elsevier, v. 63, n. 2, p. 148–167, 2012.

BLUMBERG, R.; ATRE, S. The problem with unstructured data. **Dm Review**, POWELL PUBLISHING INC, v. 13, n. 42-49, p. 62, 2003.

BORDIGNON, A. C. de A. et al. Natural language processing in business process identification and modeling: A systematic literature review. In: ACM. **Proceedings of the XIV Brazilian Symposium on Information Systems**. [S.l.], 2018. p. 25.

BROCKE, J. V.; ROSEMANN, M. et al. **Handbook on business process management**. [S.l.]: Springer, 2010.

BUSEMANN, S. Best-first surface realization. **arXiv preprint cmp-lg/9605010**, 1996.

CAPORALE, T. A tool for natural language oriented business process modeling. In: **ZEUS**. [S.l.: s.n.], 2016. p. 49–52.

CHANNABASAVAIAH, K.; HOLLEY, K.; TUGGLE, E. Migrating to a service-oriented architecture. **IBM DeveloperWorks**, v. 16, p. 727–728, 2003.

DIJKMAN, R. et al. Similarity of business process models: Metrics and evaluation. **Information Systems**, Elsevier, v. 36, n. 2, p. 498–516, 2011.

DIJKMAN, R.; DUMAS, M.; GARCÍA-BAÑUELOS, L. Graph matching algorithms for business process model similarity search. In: SPRINGER. **International conference on business process management**. [S.l.], 2009. p. 48–63.

DUMAS, M.; GARCÍA-BAÑUELOS, L.; DIJKMAN, R. M. Similarity search of business process models. **IEEE Data Eng. Bull.**, v. 32, n. 3, p. 23–28, 2009.

DUMAS, M.; HOFSTEDE, A. H. T. Uml activity diagrams as a workflow specification language. In: SPRINGER. **International conference on the unified modeling language**. [S.l.], 2001. p. 76–90.

DUMAS, M. et al. **Fundamentals of business process management**. [S.l.]: Springer, 2013.

DUMAS, M. et al. Fundamentals of business process management. In: . [S.l.]: Springer, 2018.

ELHADAD, M.; ROBIN, J. Surge: a comprehensive plug-in syntactic realization component for text generation. **Computational Linguistics**, Citeseer, v. 99, n. 4, 1997.

ELSTERMANN, M.; HEUSER, T. Automatic tool support possibilities for the text-based s-bpm process modelling methodology. In: ACM. **Proceedings of the 8th International Conference on Subject-oriented Business Process Management**. [S.l.], 2016. p. 3.

EPURE, E. V. et al. Automatic process model discovery from textual methodologies. In: IEEE. **Research Challenges in Information Science (RCIS), 2015 IEEE 9th International Conference on**. [S.l.], 2015. p. 19–30.

ERL, T. **Soa: principles of service design**. [S.l.]: Prentice Hall Upper Saddle River, 2008.

FERRARI, A. et al. Improving the quality of business process descriptions of public administrations: Resources and research challenges. **Business Process Management Journal**, Emerald Publishing Limited, v. 24, p. 49–66, 2017.

FERREIRA, R. C. B.; THOM, L. H.; FANTINATO, M. A semi-automatic approach to identify business process elements in natural language texts. In: **Proceedings of the 19th International Conference on Enterprise Information Systems. To appear**. [S.l.: s.n.], 2017.

FERREIRA, R. C. B. et al. Assisting process modeling by identifying business process elements in natural language texts. In: SPRINGER. **International Conference on Conceptual Modeling**. [S.l.], 2017. p. 154–163.

FIALLI, J.; VAJJHALA, S. The java architecture for xml binding (jaxb). **JSR Specification, January**, 2003.

FIELDING, R. T.; TAYLOR, R. N. **Architectural styles and the design of network-based software architectures**. [S.l.]: University of California, Irvine Doctoral dissertation, 2000.

FRIEDRICH, F. Automated generation of business process models from natural language input. **M. Sc., School of Business and Economics. Humboldt-Universität zu Berli**, Citeseer, 2010.

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.

GHOSE, A.; KOLIADIS, G.; CHUENG, A. Process discovery from model and text artefacts. In: IEEE. **Services, 2007 IEEE Congress on**. [S.l.], 2007. p. 167–174.

GOLDBERG, E.; DRIEDGER, N.; KITTREDGE, R. I. Using natural-language processing to produce weather forecasts. **IEEE Intelligent Systems**, IEEE, n. 2, p. 45–53, 1994.

GONCALVES, J. C. de A.; SANTORO, F. M.; BAIAO, F. A. Business process mining from group stories. In: IEEE. **Computer Supported Cooperative Work in Design, 2009. CSCWD 2009. 13th International Conference on**. [S.l.], 2009. p. 161–166.

GONÇALVES, J. C. de A.; SANTORO, F. M.; BAIÃO, F. A. A case study on designing business processes based on collaborative and mining approaches. In: IEEE. **Computer Supported Cooperative Work in Design (CSCWD), 2010 14th International Conference on**. [S.l.], 2010. p. 611–616.

GRIGORI, D. et al. Ranking bpel processes for service discovery. **IEEE Transactions on Services Computing**, IEEE, v. 3, n. 3, p. 178–192, 2010.

HADLEY, M. J. Web application description language (wadl). Sun Microsystems, Inc., 2006.

HALLERBACH, A.; BAUER, T.; REICHERT, M. Managing process variants in the process lifecycle. 2008.

HEARST, M. A. Multi-paragraph segmentation of expository text. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 32nd annual meeting on Association for Computational Linguistics**. [S.l.], 1994. p. 9–16.

HEARST, M. A. Texttiling: Segmenting text into multi-paragraph subtopic passages. **Computational linguistics**, MIT Press, v. 23, n. 1, p. 33–64, 1997.

HEINONEN, O. Optimal multi-paragraph text segmentation by dynamic programming. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 2**. [S.l.], 1998. p. 1484–1486.

HERBST, J.; KARAGIANNIS, D. An inductive approach to the acquisition and adaptation of workflow models. In: **Proceedings of the IJCAI**. [S.l.: s.n.], 1999. v. 99, p. 52–57.

HEUSER, T.; ELSTERMANN, M. Working with natural language texts for process management: Proposal and analysis of a process analysis methodology. In: ACM. **Proceedings of the 8th International Conference on Subject-oriented Business Process Management**. [S.l.], 2016. p. 2.

IORDANSKAJA, L. et al. Generation of extended bilingual statistical reports. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 14th conference on Computational linguistics-Volume 3**. [S.l.], 1992. p. 1019–1023.

JIANHONG, Y.; SONG, W. Transformation of bpmn diagrams to yawl nets. **Journal of Software**, Citeseer, v. 5, n. 4, p. 396–404, 2010.

JURAFSKY, D.; MARTIN, J. H. **Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition**. [S.l.]: Prentice Hall, Pearson Education International, 2009. 1–1024 p.

KUMARI, S.; RATH, S. K. Performance comparison of soap and rest based web services for enterprise application integration. In: IEEE. **Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on**. [S.l.], 2015. p. 1656–1660.

LANZ, A.; KOLB, J.; REICHERT, M. Enabling personalized process schedules with time-aware process views. In: SPRINGER. **International Conference on Advanced Information Systems Engineering**. [S.l.], 2013. p. 205–216.

LAVOIE, B.; RAMBOW, O. A fast and portable realizer for text generation systems. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the fifth conference on Applied natural language processing**. [S.l.], 1997. p. 265–268.

LAVOIE, B.; RAMBOW, O.; REITER, E. The modelexplainer. In: **Eighth International Natural Language Generation Workshop (Posters and Demonstrations)**. [S.l.: s.n.], 1996.

LEOPOLD, H. **Natural language in business process models**. Thesis (PhD) — Springer, 2013.

LEOPOLD, H. et al. Integrating textual and model-based process descriptions for comprehensive process search. In: **Enterprise, Business-Process and Information Systems Modeling**. [S.l.]: Springer, 2016. p. 51–65.

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

LEOPOLD, H.; MENDLING, J.; POLYVYANYY, A. Supporting process model validation through natural language generation. **IEEE Transactions on Software Engineering**, IEEE, v. 40, n. 8, p. 818–840, 2014.

LEOPOLD, H.; SMIRNOV, S.; MENDLING, J. Refactoring of process model activity labels. In: SPRINGER. **International Conference on Application of Natural Language to Information Systems**. [S.l.], 2010. p. 268–276.

LI, J. et al. A policy-based process mining framework: mining business policy texts for discovering process models. **Information Systems and E-Business Management**, Springer, v. 8, n. 2, p. 169–188, 2010.

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.

MAQBOOL, B. et al. A comprehensive investigation of bpmn models generation from textual requirements—techniques, tools and trends. In: SPRINGER. **International Conference on Information Science and Applications**. [S.l.], 2018. p. 543–557.

MARNEFFE, M.-C. D. et al. Generating typed dependency parses from phrase structure parses. In: GENOA ITALY. **Proceedings of LREC**. [S.l.], 2006. v. 6, n. 2006, p. 449–454.

MAYER, R. J. et al. **Information integration for concurrent engineering (IICE) IDEF3 process description capture method report**. [S.l.], 1995.

MCROY, S. W.; CHANNARUKUL, S.; ALI, S. S. Text realization for dialog. In: **Proceedings of the International Conference on Intelligent Technologies**. [S.l.: s.n.], 2000.

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.

MEZIANE, F.; ATHANASAKIS, N.; ANANIADOU, S. Generating natural language specifications from uml class diagrams. **Requirements Engineering**, Springer, v. 13, n. 1, p. 1–18, 2008.

MILLER, G. A. Wordnet: a lexical database for english. **Communications of the ACM**, ACM, v. 38, n. 11, p. 39–41, 1995.

MUEHLEN, M. Z.; HO, D. T.-Y. Risk management in the bpm lifecycle. In: SPRINGER. **International Conference on Business Process Management**. [S.l.], 2005. p. 454–466.

MURATA, T. Petri nets: Properties, analysis and applications. **Proceedings of the IEEE**, IEEE, v. 77, n. 4, p. 541–580, 1989.

NATSCHLÄGER, C. Towards a bpmn 2.0 ontology. In: SPRINGER. **International Workshop on Business Process Modeling Notation**. [S.l.], 2011. p. 1–15.

NAWROCKI, J. R. et al. Describing business processes with use cases. In: **BIS**. [S.l.: s.n.], 2006. p. 13–27.

NURSEITOV, N. et al. Comparison of json and xml data interchange formats: a case study. **Caine**, v. 9, p. 157–162, 2009.

OMG. Business process modeling notation (BPMN). version 2.0.2, 2013. Available from Internet: <https://www.omg.org/spec/BPMN/>.

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.

PAUTASSO, C.; WILDE, E. Why is the web loosely coupled?: a multi-faceted metric for service design. In: ACM. **Proceedings of the 18th international conference on World wide web**. [S.l.], 2009. p. 911–920.

PAUTASSO, C.; ZIMMERMANN, O.; LEYMANN, F. Restful web services vs. big'web services: making the right architectural decision. In: ACM. **Proceedings of the 17th international conference on World Wide Web**. [S.l.], 2008. p. 805–814.

PITTKE, F. et al. Context-sensitive textual recommendations for incomplete process model elements. In: SPRINGER. **International Conference on Business Process Management**. [S.l.], 2015. p. 189–197.

POLYVYANYY, A.; VANHATALO, J.; VÖLZER, H. Simplified computation and generalization of the refined process structure tree. In: SPRINGER. **International Workshop on Web Services and Formal Methods**. [S.l.], 2010. p. 25–41.

REITER, E.; DALE, R. Building applied natural language generation systems. **Natural Language Engineering**, Cambridge University Press, v. 3, n. 1, p. 57–87, 1997.

REITER, E.; DALE, R. **Building natural language generation systems**. [S.l.]: Cambridge university press, 2000.

REN, M.; LYYTINEN, K. J. Building enterprise architecture agility and sustenance with soa. **Communications of the Association for Information Systems**, v. 22, n. 1, p. 4, 2008.

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.

ROSPOCHER, M.; GHIDINI, C.; SERAFINI, L. An ontology for the business process modelling notation. In: **FOIS**. [S.l.: s.n.], 2014. p. 133–146.

RUSSELL, N. et al. Workflow control-flow patterns: A revised view. **BPM Center Report BPM-06-22, BPMcenter. org**, p. 06–22, 2006.

SÀNCHEZ-FERRERES, J.; CARMONA, J.; PADRÓ, L. Aligning textual and graphical descriptions of processes through ilp techniques. In: SPRINGER. **International Conference on Advanced Information Systems Engineering**. [S.l.], 2017. p. 413–427.

SCHULTHEISS, L. A.; HEILIGER, E. M. Techniques of flow-charting. **Clinic on Library Applications of Data Processing (1st: 1963)**, Graduate School of Library Science. University of Illinois at Urbana-Champaign, 1963.

SCHUMACHER, P.; MINOR, M.; SCHULTE-ZURHAUSEN, E. Extracting and enriching workflows from text. In: IEEE. **Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on**. [S.l.], 2013. p. 285–292.

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.

VAKULENKO, S. **Extraction of Process Models from Business Process Descriptions**. [S.l.]: Citeseer, 2011.

VANHATALO, J.; VÖLZER, H.; KOEHLER, J. The refined process structure tree. **Data & Knowledge Engineering**, Elsevier, v. 68, n. 9, p. 793–818, 2009.

WESKE, M. Business process management architectures. In: **Business Process Management**. [S.l.]: Springer, 2012. p. 333–371.

ZIMMERMANN, O. et al. Second generation web services-oriented architecture in production in the finance industry. In: ACM. **Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications**. [S.l.], 2004. p. 283–289.

ZIMOCH, M. et al. The repercussions of business process modeling notations on mental load and mental effort. 2018.