BPMN2TEXT: A Language-Independent Framework to Generate Natural Language Text from BPMN models

Raphael de Almeida Rodrigues[1], **Leonardo Guerreiro Azevedo**[1,2], **Kate Revoredo**[1]

[1]Post Graduate Program in Informatics (PPGI)
Federal University of the State of Rio de Janeiro (UNIRIO)
Av. Pasteur, 456 – Urca – Rio de Janeiro – RJ – Brazil – 22290-240

[2]IBM Research - Brazil
Av. Pasteur 146 – Botafogo – Rio de Janeiro – RJ – Brazil – 22290–240

{raphael.rodrigues, azevedo, katerevoredo}@uniriotec.br,

LGA@br.ibm.com

Abstract. *Graphical representation of business processes helps better understanding the process, and consequently improves their execution. BPMN is the most used notation for business process modeling. Domain specialists, which are experts in the business, usually do not have the modeling skills (e.g., knowledge about the BPMN notation) to easily read and understand business process models. Reading a natural language text that represents a model is easier for the specialists to understand the process than reading the model. This work proposes a language-independent framework aiming at automatically generating natural language texts from BPMN business process models. The framework was implemented using open-standard technologies and it was used to develop tools able to generate text from BPMN models written in English and in Portuguese. An experiment was conducted to evaluate the equivalence between the text and model representations. As a result, the textual work instructions can be considered equivalent, in terms of knowledge representation, to process models represented in BPMN. Regarding the quality of the generated textual descriptions, 86% of the participants claim it varies from excellent to good.*

Resumo. *A representação gráfica de um processo de negócio ajuda no entendimento do processo e, consequentemente, melhora sua execução. BPMN tem sido usado como a notação padrão para os modelos de processos de negócio. Especialistas do domínio, que são especialistas no negócio, não têm as habilidades de modelagem necessárias para ler com facilidade um modelo de processo de negócio. Para eles, é mais fácil ler o processo representado em linguagem natural. Neste trabalho, propomos um framework independente de idioma que gera automaticamente textos em linguagem natural a partir de modelos de processos de negócio elaborados usando BPMN. O framework foi implementado usando tecnologias abertas e foi usado para desenvolver ferramentas capazes de gerar texto de modelos BPM escritos em inglês e em português. Um experimento foi realizado para avaliar a equivalência entre as representações em texto e modelo. Como resultado, as instruções textuais podem ser consideradas equivalentes aos modelos de processo representados em BPMN em termos de representação do conhecimento. Em relação à qualidade do texto gerado pelo framework, 86% dos participantes alegam que esta varia entre excelente e bom.*

RODRIGUES, R. A.; AZEVEDO, L. G.; REVOREDO, K.
BPMN2TEXT: A Language-Independent Framework to Generate Natural Language Text from BPMN models
iSys | Revista Brasileira de Sistemas de Informação, Rio de Janeiro, vol. 9, No. 4, pp. 38-56, 2016

1. Introduction

Business process models provide an abstract graphical view of organizational procedures by reducing the complex reality of the work performed by a company to its most important activities. They help to foster an understanding of the underlying organizational procedures, provide process documentation, and represent an important asset for organizational process redesign [Larman 2005, Kettinger et al. 1997]. However, the validation and usage of process models are often hampered because many domain experts are unable to fully understand these models since usually they are not experts in process modeling and/or the modeling notation. In fact, studies present how hard is the comprehension of process models even for those who are familiar with process modeling [Mendling et al. 2012]. Domain experts usually do not have the necessary skills to read the process models designed by business analysts [Dumas et al. 2013]. Training employees in understanding process models is costly and can hardly be considered an option for the entire workforce of a company. On the other hand, natural language texts that represent the process models are a natural alternative to business process models. In addition, discussions based on text tend to be more productive than discussions based on models [Castro et al. 2011].

This work proposes a language-independent framework aiming at automatically generating natural language texts from BPMN business process models. Hence, the framework can be used to implement tools able to read models written in languages (like Portuguese, English, Spanish) and generate corresponding texts. The framework was implemented using open-standard technologies (*e.g.*, Java) and it was used to create tools able to generate text from BPMN models written in English and in Portuguese.

In a previous work, we presented a tool implemented using the first version of the framework [Rodrigues et al. 2014] to generate texts from models written in English and Portuguese. In the present work, we further improve the formalism and the architecture of the framework. Existing algorithms were improved and new ones were developed, *e.g.*, bugs were fixed, the translation of other BPMN elements was implemented and existing translations were improved. Moreover, experiments were executed to evaluate the quality of the framework's output (generated text) and to assert whether the generated text is capable of transmitting the same knowledge as compared with a business process model.

To the best of our knowledge, we are the first to propose a tool for generating natural language text from business process models that can be adapted to a wide range of languages. There are, however, tools that address the opposite direction, *i.e.*, there are works on generating models (process models, ontologies and UML diagrams) from natural language text [Friedrich et al. 2011, Leão et al. 2013, Bajwa and Choudhary 2006]. While all these techniques address different challenges, they mainly differ from our technique by using real natural language text as input. The main challenge for generating text from process models, which is addressed by our framework, is to adequately analyze the existing natural language fragments from the process model elements, and to organize the information from the process model in a sequential fashion.

The remainder of this work is structured as follows. Section 2 presents the research methodology applied in this work. Section 3 presents the main concepts about NLG (Natural Language Generation). Section 4 presents the proposal to generate text in natural language. Section 5 presents the framework's evaluation. Section 6 presents the related work. Finally, Section 7 presents the conclusion and proposals of future work.

2. Scientific Research Methodology

We applied the Design Science methodology [Henver et al. 2004] in four steps:

- **Analyze business process models**: We collected initial test data, consisting of several process models and natural language process instructions. They provided insights on how to translate business process models to text.
- **Map text constructs to process model elements**: We analyzed linguistic semantics and syntactic patterns to create the mappings. They revealed issues regarding the quality of the syntax parser and the text itself. We systematically categorized these issues and developed a conceptual solution strategy.
- **Transformation rules definitions**: We defined rules using heuristics which were iteratively refined. The rules were implemented in the research prototype.
- **Evaluation**: We evaluated the implemented framework against several process models and gathered important information about its behavior. This information was used in another iteration to refine and discover new patterns and rules, consequently improving the algorithms.

The proposed framework was evaluated using a qualitative and quantitative approach in two ways.

First, a controlled experiment was executed on artificial and real process models. Ten (10) different business process models in English and in Portuguese were designed by the authors to stress specific conditions (scenarios) that should be covered by the framework. Twenty (20) business process models were gathered from universities and companies. Afterwards, the framework was executed over these models, and the generated texts were compared with the models, assessing: (i) If the generated text is capable to represent the same knowledge the process model represents; (ii) How far the generated texts are distant from manually created texts according to some text metrics.

Second, an evaluation was conducted using parts of the real business process models. This evaluation was an exploratory research, made-up by research questions to investigate whether the generated text is capable of transmitting the same knowledge as compared with a business process model. The research questions were presented to participants coming from industry and universities. Besides, the participants could also leave comments (*i.e.*, feedback) about the generated text.

3. Natural Language Text Generation

This section presents Reiter and Dale's generic architecture to generate natural language texts [Ehud Reiter 1997]. This architecture is widely used to develop NLG systems that must manipulate texts in natural language.

3.1. Natural Language Generation Text Pipeline

Many natural language generation systems follow a pipeline approach (Figure 1) consisting of three main steps [Ehud Reiter 1997]:

1. **`Text Planning`**: This step determines the information to be communicated in the text and defines the order which the information will be conveyed.

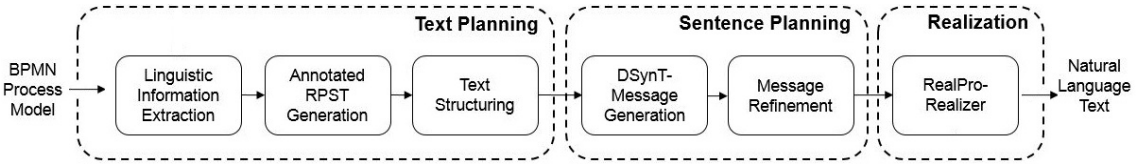

Figure 1. Three steps pipeline.

2. **`Sentence Planning`**: This step chooses specific words to express the determined information. If applicable, messages are aggregated and pronouns are introduced in order to obtain variety.
3. **`Sentence Realization`**: This step transforms the messages into grammatically correct sentences.

Reiter and Dale proposal is abstract, *i.e.*, it does not take into account technology constraints or specific algorithms to execute the defined activities of each step. This pipeline can be used as a guide to develop new systems that use NLG technologies [Ehud Reiter 1997]. It is development team's responsibility to decide which technology and algorithms are more suitable for instantiating each step of the pipeline.

Section 4 presents our framework. Several algorithms and classes were developed to incorporate the required functionality of each step.

4. Generic Framework to Generate Text from Process Models

The proposal of this work is an application framework [Pree 1994]: "*Application frameworks consist of ready-to-use and semi-finished building blocks. The overall architecture is predefined as well. Producing specific applications usually means to adjust building blocks to specific needs by overriding some methods in subclasses*" . This section presents the framework components (Section 4.1) and how the framework implementations handle each stage of the three steps pipeline to generate text in natural language (Section 4.2).

4.1. Framework Components

The framework is composed by several ready-to-use building blocks (Frozen spots) and defines interfaces (Hot spots) which must be implemented for each specific language [Pree 1994] (Figure 2) . Therefore, the developer does not need to know in details how the NLG process (*i.e.*, pipeline NLG) works, but rather understand what should be implemented to fit the hot spots. The components of this model are described as follows.

- *GeneralLanguageCommon* is a generic (language-independent) module. It includes interfaces' definitions which must be implemented for a specific language. It contains the necessary infrastructure to execute the NLG pipeline process. It includes the data structures, and it knows exactly which and when an object must be called to deal with a specific phase of the pipeline. For example, regarding the *Localization strategy* (represented by the classes of the `Localization` package), the module knows when to call the `LocalizationManager` object to translate a specific message, retrieved from the `LocalizationMessages` enumeration (keys) during the text information extraction. *E.g.*, for the key *PROCESS_BEGIN_WHEN*, the returned text would be "*O processo começa quando*"

for Portuguese. Analogously, it knows when trigger each implemented interface method for a given language at runtime. The textttGeneralLanguageCommon is composed by:

  - *LabelAnalysis* contains all interfaces for label manipulation. This package must be implemented for each supported language, *i.e*, all the linguistic classification algorithms must be implemented for each language. *E.g.*, algorithms to classify words into verbs, nouns, adjectives. In particular, it is capable of parsing labels written using the most common naming conventions (*e.g.*, label styles [Leopold et al. 2013].
  - *Localization* contains all classes to execute words localization logic. *E.g.*, the `LocalizationManager` class is used to fetch messages which will be used in the final text representation. *E.g.*, the message "*Process begin when*" is automatically inserted into the beginning of an English natural language text. If the model's language is Portuguese, the message would be "*O processo começa quando*". This class is language independent. Hence, the developer does not have to change or override its implementation.
  - *DSynt* contains all classes used to represent DSynts trees (Section 4.2). DSynt has the same representation for any language.
  - *Fragments* contains all classes to represent a sentence in natural language pattern. This data structure is the same for any language.
  - *ISurfaceRealizer* defines what language-specific realizers must be implemented to produce messages in a natural language format.

- *LanguageRealizer* contains the classes for the concrete implementation of interfaces and abstract classes defined in `GeneralLanguageCommon` package for each language (*e.g.*, `EnglishRealizer`, `PortugueseRealizer`).
- *MultiLanguageProject/LanguageConfig* works as a Factory[1], instantiating classes' objects that implement the interfaces for language-specific implementations.

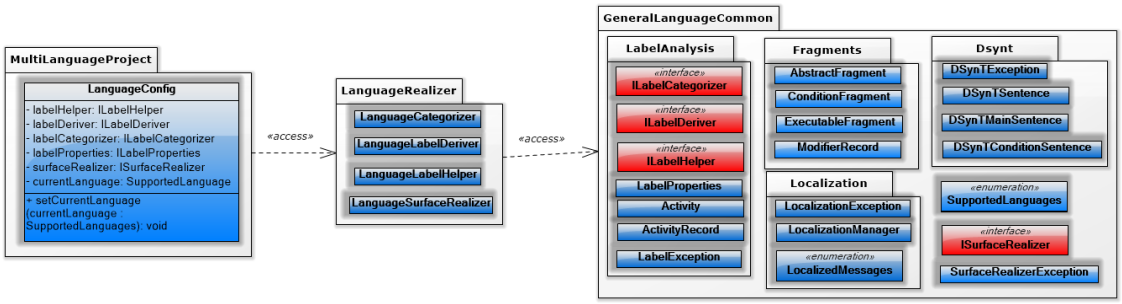

Figure 2. Framework packages used to generate text from business process models.

4.2. Framework pipeline instantiation

This section details how the three-steps of NLG pipeline process, presented in Figure 1, is handled by the framework.

[1] A factory is a program component which main responsibility is the creation of other objects

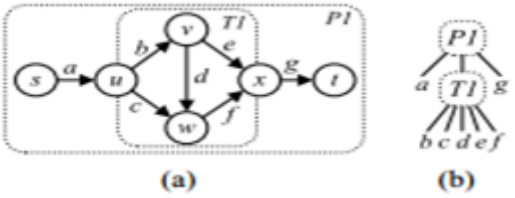

Figure 3. (a) A graph and its canon fragments, (b) RPST tree that represents the graph (a).

4.2.1. Text Planing

The following implementations handle the *Text Planing* step:

- *Linguistic Information Extraction*: The package `LabelAnalysis` of the `GeneralLanguageCommon` module contains all methods that are used during this step. The implemented components infer the linguistic information from all the labels (*e.g.*, name of activities, description, event labels, gateways labels and actors) of a business process model. In a nutshell, the framework reads all labels information from the process model to instantiate `LabelProperties` objects, *e.g.*, actor, action, business object etc. The identification of the semantic function of a specific word is given by the `ILabelHelper` implementation of the specific language (*e.g.*, Portuguese or English). The framework also generates a graph that represents the business process model.
- *Annotated RPST generation*: In this step, the framework creates a RPST tree representation using the process model graph representation. RPST (*Refined Process Structure Tree*) is a tree that contains a hierarchy of sub graphs derived from the original graph [Polyvyanyy et al. 2011]. The RPST is based on observing that every graph can be decomposed in a logically independent subgraph hierarchy having only one input and only one output, which are called fragments. In one RPST any of these two fragments are interconnected or independents. The resulting hierarchy can be seen as a tree where the root of the tree and the leaves are fragments with a single arch. Besides creating the tree, the framework tags its nodes with text formatting information, *e.g.*, indication of paragraphs or bullets, indentation level, use of comma etc. A RPST sample is depicted in Figure 3.
- *Text Structuring*: In this step, the framework traverses the RPST tree nodes and set structuring properties which are used to present the text in a fashion format and improving the text's quality and readability. For example, if an activity must start a new paragraph, the attribute *hasParagraph* of the respective node is set to true. Others properties, like *hasBullet* and *senLevel* are used to help with the indentation of the final text. For example, these properties are read by the `Message Realization` component to generate the text with the correct indentation for activities that are executed in parallel.

4.2.2. Sentence Planning

The following implementations handle the *Sentence Planning* step:

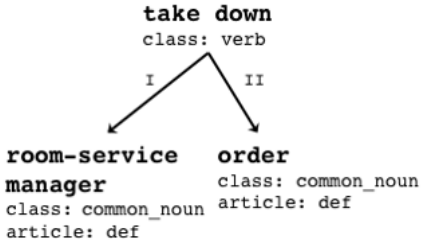

Figure 4. Example of DSynT tree.

- *DSynT Message Generation*: In this step, the framework transforms the previously created RPST tree into a list of intermediate messages. In other words, the linguistic information of the model is not directed mapped to the final text, rather it is mapped to a conceptual representation that is still suitable for changes. In particular, each sentence is stored into a DSynt tree.
  A DSynT tree (Deep-Syntatic tree) is a dependency representation introduced by the Meaning Text Theory [Mel'čuk and Polguere 1987]. In a DSynT tree, each node is labeled with a lexeme semantically complete, *i.e.*, lexemes as conjunctions or auxiliary verbs are excluded. Besides, each lexeme has grammatical information about themselves (metadata). For instance, voice and conjunction of the verbs or number and substantive definition. The advantages in using DSynT trees are: it is rich, but still changeable; it represents sentences, and there are tools that, given a DSynT tree as input, are capable of changing it directly in a grammatical correct sentence. Figure 4 presents an example of a DSynT tree of the activity "*Take down the order*" executed by the "*room-service manager*".
  The Dsynt package has all the classes that are used in this phase.
- *Message Refinement*: This step is responsible to the message aggregation task. The need for message aggregation arises when the process contains a big sequence of activities. In this case, we can use three types of aggregation techniques: aggregation by actor; aggregation by business object; and, aggregation by action. There are two classes involved in the aggregation task: `SentenceAggregator`, which aggregates sentences executed by the same actor (role) and `ReferringExpressionAggregator`, which aggregates sentences through the addition of referring expressions, *i.e.*, it adds pronouns to avoid unnecessary repetition of actors.

4.2.3. Sentence Realization

This section details the last step of the NLG pipeline where the framework generates the natural language text that represents the process model. The `ISurfaceRealizer` interface of `GeneralLanguageCommon` package defines which methods should be implemented to handle a specific language. This interface presents general operations of languages belonging to the Romanian and Germanic sub-branches of the Indo-European language family. In a nutshell, through the implementation of the methods's interface, specific language implementations are able to read textual information from the nodes of DSynT trees and assemble grammatically correct sentences. This process is triggered by

the `realizeSentence` method of `ISurfaceRealizer` interface).

5. Evaluation

This section presents the evaluation of the proposed framework through the implemented prototype. First, a controlled experiment was designed and executed by the authors with a small set of business process models to evaluate the framework's behavior. Afterwards, a second evaluation was conducted through an exploratory research to investigate whether the generated text is capable of transmitting the same knowledge as compared with the corresponding business process model.

5.1. Controlled Experiment

This evaluation validates the behavior of the framework using artificial and real data.

For the artificial data, ten (10) artificial business process models[2] were designed by the authors to stress specific conditions that should be covered by the framework. The framework's behavior was observed in different scenarios: process models composed by activities without any control flow logic and process models composed by activities with control flow logic (*e.g.*, gateways AND, XOR and parallel). These models presented variations of important characteristics for the evaluation, which were:

- **Activities being executed in sequence by the same actor (role)**: These activities should activate the modules for referring expression and message aggregation.
- **Activities without roles (empty lane)**: These activities test if activities without actors are correctly mapped to a textual description using passive voice.

Besides, as the framework was designed to support multiple languages, the set of the artificial process models were designed in both English and Portuguese. With this strategy, it was also possible to test if the framework could deal with a dynamic language change in the process model during execution time, initializing the necessary configurations to trigger the right implementations to extract the process model elements semantic. The language detection is identified through a meta-data in the process model, which informs the language used to design it.

In the case of real data, a set of twenty (20) business process models[3] were gathered from universities and companies to stress real scenarios. These process models were written in Portuguese. Hence, each model was translated also to English for testing the language-independence framework's feature.

In total, thirty (30) process models were used during the controlled evaluation. The overall characteristics of these models are presented in Table 1. Each process contained at least one (1) actor and no more than six (6) actors. In average, the whole data set contained 2.96 actors per process. Analogously, each process contained at least two (2) activities and no more than forty (40) activities. In average, the whole data set contained 11.68 activities per process. Finally, not all processes have control flow logic and ten (10) is the maximum of control flows a process has. In average, each process contained 3.48 control flows.

[2]The set of process models (artificial data) are available at `https://bitbucket.org/rar150/unirio-workspace/src`

[3]Due to copyright reasons, the whole set is not available. Nevertheless, some process models are available at `https://bitbucket.org/rar150/unirio-workspace/src`

Table 1. Overall characteristics of the complete test data set.

Metric	MIN	MAX	Average
Number of Actors	1	6	2.96
Number of Activities	2	40	11.68
Number of Control Flows	0	10	3.48

The evaluation was conducted in the following steps:

- **Step 1**: The test begun by fetching a process model from the repository.
- **Step 2**: Then, the framework was used to generate the text of the process model.
- **Step 3**: We compared the output text with the model, and investigated divergences.
- **Step 4**: If any error was found, the framework was improved to fix the it. After applying the fixes, the process model was resubmitted to the framework (Step 2). This procedure was repeated until there was no critical error.
- **Step 5**: Finally, after asserting no more divergences, the test was considered to be concluded successfully.

5.2. Exploratory Evaluation

This section presents the design of the proposed experiment, including our research questions, the instruments to address these questions, the participants of the study, and the measurements take. The goals of the experiment were: (a) Analyze the equivalence between textual description and process models; (b) Analyze the textual description quality according to the participant's perspective; and, (c) Analyze how the experience influences the understanding of textual descriptions and process models.

5.2.1. Research Questions

The experiment's main objective was *Assess whether the knowledge represented by the generated process description (i.e., textual work instructions) can be considered equivalent to the process model.* Two research questions were proposed to address this issue:

1. Is the knowledge represented by the natural language text, generated by the framework, equivalent to the process model?
2. Does the natural language text, generated by the framework, need to be enhanced to achieve better understanding?

5.2.2. Instrumentation

An online questionnaire[4] was used to collect the data for this experiment. The questionnaire was composed by: (i) A set of questions to characterize participant experience in process modeling; and, (ii) A set of seven (7) Text-Model pairs (*i.e.*, seven pairs of BPMN model and corresponding textual work instructions) describing process fragments, followed by three questions. The questions aimed to: (a) Rate the equivalence between the textual work instruction and the BPMN model; (b) Evaluate the text quality; and, (iii) Receive comments from participants about the generated textual work instruction.

[4]The questionnaire is available at https://goo.gl/forms/KExDuz2sph7swVx03

Process fragments were chosen instead of whole processes in order to minimize the time required for the participants to fill up the whole questionnaire. A long time response questionnaire would lead to fewer participants and could compromise the study quality (*i.e.*, whole process could not be comprehensible in a short time span).

The experiment was executed in several sessions, each involving a subset of the participants. The overall structure of a session was comprised by three (3) steps. First, the questionnaire was electronically sent through e-mail. Next, the questionnaire was filled by the participants. Finally, the answers were collected and stored in the database. No time restriction was imposed to participants, either for analyzing the process fragment or for filling up the questionnaire.

The characteristics of the used instrument's elements are described below. They were originally written in Portuguese and all participants spoke Portuguese as their native language. For the purpose of consistency with the remaining text, we present English translated versions of the original elements.

Participants Characterization Questions: It was a set composed by four closed-ended questions presented in Table 2. Questions were answered according to a 5-point ordinal scale (*i.e.*, likert scale) list. These questions were elaborated to obtain detailed and specific information about the participant's experience related to process modeling and BPMN notation, avoiding subjective answers. The ordinal scale was used to specify the participant's level of agreement or disagreement with a series of statements about business process modeling. Ordinal scales (usually with five or seven symmetrical points) are frequently used in survey research and are a reliably measure mental effort [Paas et al. 2003].

Table 2. Questions about the participant's experience in process models.

Questions and Options
1 - Overall, I am very familiar with process modeling.
a. Strongly disagree | b. Slightly disagree | c. Neutral | d. Slightly agree | e. Strongly agree
2 - Overall, I am very familiar with the BPMN notation.
a. Strongly disagree | b. Slightly disagree | c. Neutral | d. Slightly agree | e. Strongly agree
3 - I feel very confident in understanding BPMN process models.
a. Strongly disagree | b. Slightly disagree | c. Neutral | d. Slightly agree | e. Strongly agree
4 - I feel very competent in using BPMN for process modeling.
a. Strongly disagree | b. Slightly disagree | c. Neutral | d. Slightly agree | e. Strongly agree

Process Fragment (Text-Model Pair): The text-model pair aimed at rating the equivalence in terms of the information transmitted by both. Each pair was composed by a BPMN model and the corresponding textual work instructions generated by the framework. Figure 5 depicts one exemplary of the text-model pair used by this experiment. A process fragment was presented in order to minimize the time and to abstract the need for understanding the whole process semantic. Hence, isolating domain knowledge that the participant might have about the process. Thus, the user can focus on short descriptions and fewer symbols within a BPMN model. The process pair fragment was accompanied by three questions. The first question evaluated whether the participant consider both knowledge representations able to transmit the same information about the process fragment (Table 3). The second question aimed at evaluating the textual description quality

(Table 4). Both questions were answered according to a 5-point ordinal scale. Finally, the third question was optional and answered as a free text. This question aimed at gathering qualitative data to enable an open exploratory research about the comparison between the text-model pair.

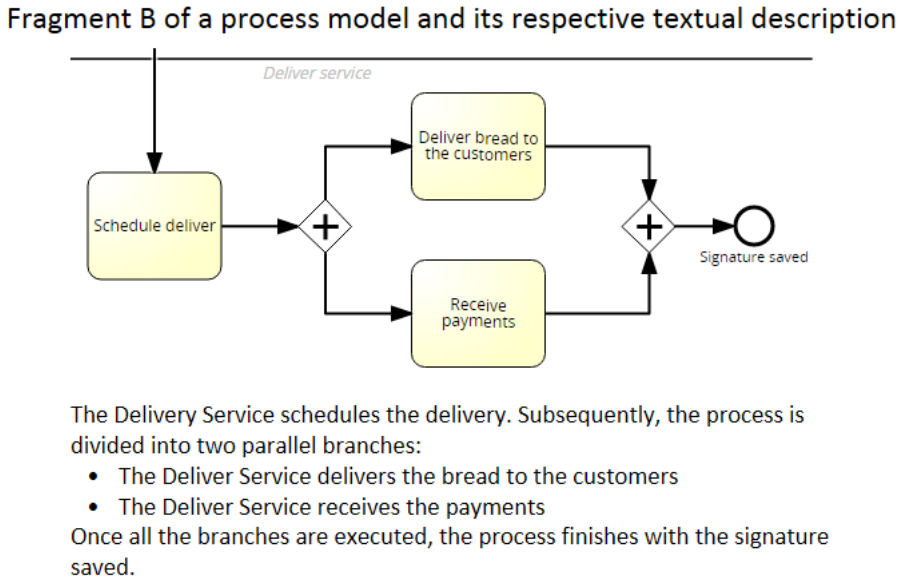

Figure 5. A text-model pair describing a process fragment.

Table 3. Question about the equivalence between the textual work instruction and the BPMN model, which describes a process fragment.

Question and Options
Both, text and model, are considered equivalent in terms of the process which they describe.
a. Totally disagree (0% equivalent)
b. Slightly disagree (Equivalence between 1 and 33%).
c. Neutral (Equivalence between 34 and 67%).
d. Slightly agree (Equivalence between 68 and 99%).
e. Totally agree (100% equivalent).

5.2.3. Participants

In total, 66 participants were selected to participate (9 students, 8 professors and 49 practitioners). Students and professors were from Federal University of the State of Rio de Janeiro (UNIRIO), State University of Rio de Janeiro (UERJ) and Federal University of Rio de Janeiro (UFRJ)). Practitioners were from different IT companies located in Rio de Janeiro (Brazil). The experience of the participants were characterized based on their answers to the characterization questions presented in Table 2.

Table 4. Question about the textual work instruction quality, which describes a process fragment.

Question and Options
How would you rate the quality of the textual description, varying from 1 to 5 (where 1 stands for horrible and 5 excellent)?
1. (horrible) | 2. (very bad) | 3. (acceptable) | 4. (very good) | 5. (excellent)

5.2.4. Measurements

Process Equivalence Measurement (RQ1): The accuracy of the research question 1 was measured by the number of correct answers. Due to the text being sensible to an open interpretation, we did not expected both representations to be 100% equivalent. Instead, we claim a threshold varying from 70% to 100% can be considered as a good result for the sake of this analysis.

We sum the quantity of answers given to the same option, regardless of the process fragment being analyzed. This allow us to have a general overview of the evaluation. Nevertheless, a secondary analysis, which considered answers only to the same process fragment, was made and is described in details in Section 5.4. In the instrument, "*Totally disagree*" and "*Totally agree*" options were considered equivalent to extreme points (0 and 100, respectively). Hence, we grouped the answers for options "*Totally disagree*" (0% equivalent) and "*Slightly disagree*" (Equivalence between 1 and 33%) into one, which give us an equivalence rating ranging from 0 to 33%. Analogously, we did the same for the options "*Slightly agree*" (Equivalence between 68 and 99%) and "*Totally agree*" (100% equivalent), which give us an equivalence rating ranging from 68 to 100%. Doing so, we shorten our analysis into three groups. The optimal scenario would be all answers within the first group (which vary from 68% to 100%) and the worst case scenario would be all answers within the third group (which vary from 0% to 33%). The counting process was performed by a specialized software from Google, known as Google Forms[5].

Another instrument we used was making the third text-model pair not equivalent. So, for this particular fragment, we expected more participants would answer that both representations were not equivalent regarding the knowledge they represent.

Textual Description Quality Measurement (RQ2): To measure the result for RQ2, we applied the same strategy used to measure RQ1 (*i.e.*, sum the quantity of answers given to the same option). Thus, to enable a better reading of the results, we grouped the answers for options "*Very Bad*" and "*Horrible*" into one group. All the remaining answers (options "*Good*", "*Very Good*" and "*Excellent*") were grouped into a second group.

We considered the reverse strategy for the third fragment, which had errors inputted on purpose. So for this particular fragment, we grouped answers between 1 and 2 with the others answers between 5 and 4 from the others fragments. Analogously, we did the same for answers between 3 and 5, which was grouped together with answers between 3 and 1 from the others fragments. In other words, for the third fragment we mapped "*Horrible*" and "*Very Bad*" to "*Excellent*" and "*Very Good*", respectively.

[5]For more information about Google Forms refer to https://www.google.com/forms/about/.

Our expectation (*i.e.*, the optimal scenario) was the number of answers for the textual quality description between 5, 4 and 3 (inclusive) be higher than the number of answers between 2 and 1. The worst case scenario would be the reverse, with more answers between 2 and 1 than answers between 5, 4 and 3. To be more precise, we expected that at least 70% of the answers were within the first group (answers between 5, 4 and 3), which we defined as a threshold for a great result.

5.3. Overall Evaluation

Figure 6 depicts the overall evaluation for RQ1, which had 342 answers within the equivalence group ranging from 68% to 100%. This result can be read as "*74% of the participants claim that the equivalence between both knowledge representations vary from 68% to 100%*[6]". It is a great result since the textual representation is written without any formal structure; therefore, encompassing ambiguity and open interpretations. We argue the chosen text structure can achieve the expected results, which is to transmit the process knowledge through a natural language representation. Based on this, the answer for RQ1 is: "*The knowledge represented by the natural language text, generated by the framework, can be considered equivalent to the process model.*" Hence, our expectation was achieved. The number of answers between 68% and 100% were higher than the sum of the other two groups (participants who rated equivalence between 0% and 67%).

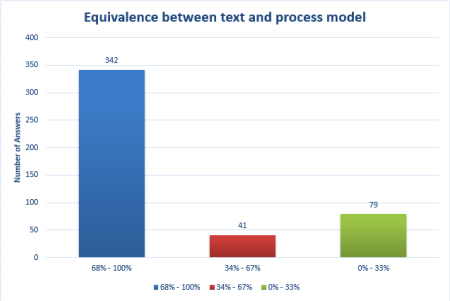

Figure 6. participant's answers distribution among equivalence intervals.

For RQ2, we calculated how many participants claim the textual quality can be classified as "*Excellent*", "*Very Good*", "*Good*", "*Very Bad*" or "*Horrible*" (Figure 7.a). Besides, we grouped answers for "*Excellent*", "*Very Good*" or "*Good*" in one group and answers for "*Very Bad*" or "*Horrible*" in another group (Figure 7.b). We expected answers for the first group were higher than the sum of all the others.

Analyzing the graphic depicted by Figure 7.b, 404 answers were within the first group (ranging from "*Good*", "*Very Good*" and "*Excellent*"). The result can be read as "*86% of the participants claim the textual description quality vary from Good, Very Good and Excellent*[7]". This can be considered a great result regarding our threshold, which was that 70% of the answers were within the first group.

[6]Each participant contributed with seven (7) answers, thus if we divide the total number of answers by seven (342/7) it give us the average number of participants that choose the same answer (48 participants)

[7]Each participant contributed with seven (7) answers, thus if we divide the total number of answers by seven (404/7), it give us the average number of participants that choose the same answer (57 participants).

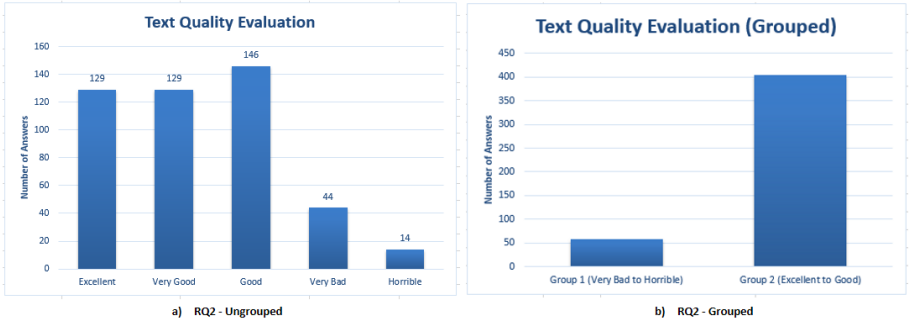

Figure 7. participant's answers distribution among: a) five groups; and b) two groups.

5.4. Isolating the analysis for each process fragment

This section presents the evaluation regarding each process fragment. Figure 8 depicts the evaluation for RQ1, while Figure 9 depicts the evaluation for RQ2.

From the results presented in Figure 8, we can state there is no big variation from one process fragment to another, with the exception for the third process fragment. As mentioned earlier (Section 5.2.2), we included errors in the third process fragment in order to check the participants confidence. So, for this fragment we expected the number of answers within the third and second group was higher that the number of answers within the first group. We got 19 answers for the first group, 3 answers for the second group and 44 answers for the third group. Thus, the participants were able to identify the flaws in the third fragment with a good accuracy (68%). If we sum the number of answers for group one for each process fragment (excluding the third fragment) with the number of answers for group three for the third fragment, and then divide the result by total number of answers, we get average of 74%, which is 4% higher then our minimal threshold (70%). Therefore, textual work instructions are able to represent the same knowledge that the model represents, within an acceptable difference due to the informal and lower abstraction given by the text format.

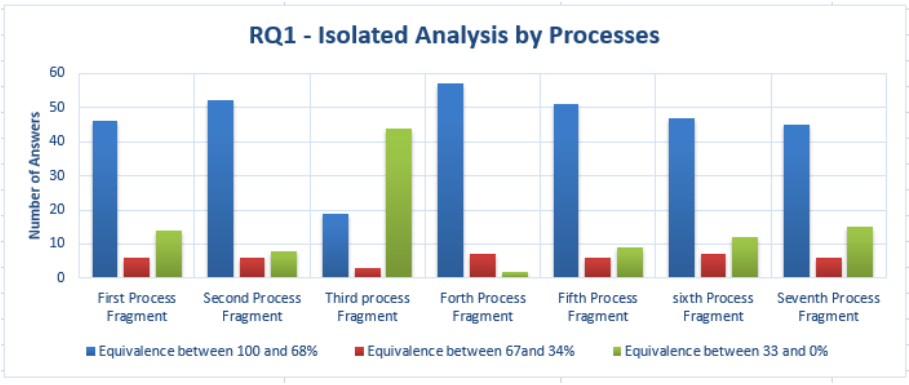

Figure 8. Chart that shows participant's answers distribution among the options available regarding the textual description quality.

From the results presented in Figure 9, we can state there is no big variation from one process fragment to another, with the exception for the third process fragment also

for RQ2. We got 41 answers within options 4 and 5, and 25 answers within options 1, 2 and 3. Thus, the participants were able to identify the flaws in the third fragment. Nevertheless, we expected the difference among these two groups were higher because it means that 38% of the participants claim the textual quality vary from good to excellent. It might have happened due to the abstraction undertaken by the participants. Most of the participants were able to abstract the model and analyze only the textual quality according to its structure. However, more data is required to assert this hypothesis. If we sum number of answers group one (number of answers for options 1, 2 and 3) for each process fragment (excluding the third fragment) with the number of answers for group two for the third fragment and then divide the result by all the seven fragments, we should have the average quality for the textual representation. The result is 86% of answers for group one, which can be read as: "*86% of the participants claim the textual description quality vary from Good, Very Good and Excellent*". This indicates the chosen textual format is good. This result is aligned with what we expected due to the use of NLG techniques (*e.g.*, Discourse Marker insertion, Referring expression generation) which are able of enhancing the text and improve its readability. We also believe the use of bullets and indentation contributed for the good evaluation.

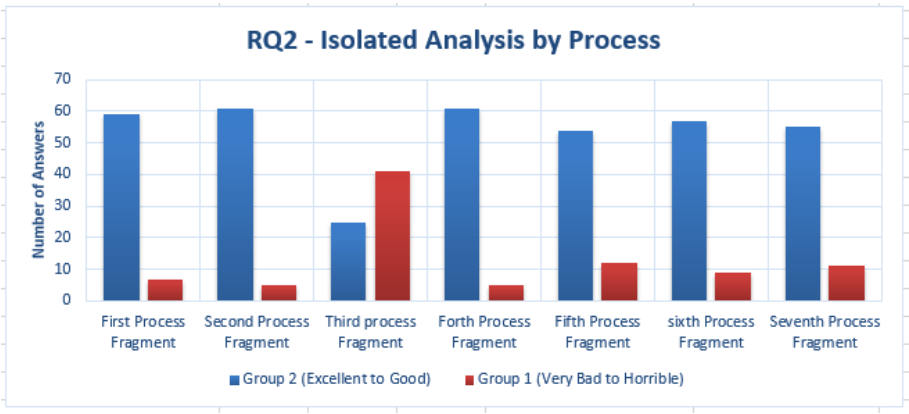

Figure 9. Chart that shows participant's answers distribution among the options available regarding the textual description quality.

6. Related Work

This section presents and compares the works related to our proposal, grouped into two specific topics: business process model understandability; and, process models to textual descriptions generation.

6.1. Business Process Model Understandability

The field of business process model understandability is discussed from different perspectives. For instance, the results from Mendling *et al.* show that the number of arcs has an important effect on the overall model understandability [Mendling et al. 2007]. In fact, many studies on process model understandability emphasize how complex the comprehension of process models can be, even for people who are familiar with process modeling [Mendling et al. 2012]. Towards addressing this problem, Leopold *et al.* (2012) propose a technique that can generate natural language text from BPMN process models, which can increase process understanding for non experienced users

[Leopold et al. 2012]. Mendling *et al.* demonstrate the impact of natural language in the activity labels for model comprehension [Mendling et al. 2010]. A more general perspective is taken by Zugal *et al.*, where the authors investigate how far the cognitive inference process affects the model understanding [Zugal et al. 2011]. The approach presented by Leopold *et al.* builds on these insights trying to lower the overall burden of process model comprehension [Leopold et al. 2014].

There is a rich body of research on the pros and cons of visual diagrams in contrast to natural language. The hypothesis that a diagram is sometimes worth ten thousand words has inspired this stream of research [Larkin and Simon 1987]. The hypothesis is based on text being limited to linear order while the spatial arrangement of different elements in a diagram allows a more efficient information processing, through inducing cognitive processes such as visual chunking, mental imagery and parallel processing [Winn 1994]. However, although the instructional and educational potential of graphical models are widely acknowledged, in some cases, they are not always more effective than other methods of representation. Usually, symbols of a graphical notations have to be learnt by readers in order to be understood [Siau 2004]. This fact is exactly what sets system analysts and domain experts aside in terms of their model readership skills [Leopold et al. 2012]. Therefore, training is required before the benefits of a graphical notation can materialize. This is supported by findings on considerable error rates in graphical process models [Mendling et al. 2006, Gruhn and Laue 2007, Mendling 2009].

The empirical findings on the strengths and weaknesses of text and diagrams are diverging. Moher *et al.* looked at several ways to express program structures in text and in Petri Nets. They state "graphics are not better than text and, in several cases, they are considerably worse" [Moher et al. 1993]. The results obtained by Shneiderman *et al.* are also aligned with Moher *et al.* findings [Shneiderman et al. 1977]. Shneiderman compared expressing program logic in flowchart and in programming language text. They found there are no statistically significant differences between the flowchart and non flowchart groups. On the other hand, in some cases, the mean scores for the non flowchart groups even surpassed the means for the flowchart groups. Finally, also aligned with the aforementioned findings, Green *et al.* compared readability of textual and graphical programming notations, and refused the hypothesis that graphics presentation would be superior. Actually, they found it is worse [Green et al. 1991].

Related work comparing process modeling notations can be roughly grouped into two categories: (i) Graphical notation comparison; (ii)Textual versus graphical notation comparison. The first is the most prominent one. Several of these studies suggest expertise is the most relevant factor in comprehension [Curtis et al. 1989], but there is no absolute better or worse representation. However, while process understanding and comprehension has been intensively studied in recent research, there is a research gap on how the comprehension of business process can be affected when the information is presented in natural language text or a process model, according to the reader's experience with process modeling.

There are also other related work that focus on comparing business process understanding using different approaches for presenting the information. For example, compare declarative process models against a text based notation using participants that have some experience in modeling declarative process [Haisjackl and Zugal 2014]. Differently to

that work, our research used imperative process models, we involved participants whose experience with process modeling vary from none to expert, and presented a natural language text simulating a human description of the process [Haisjackl and Zugal 2014].

In another perspective, Ottensooser *et al.* compare model understanding against use cases presented in text format using the Cockburn notation. We used a more fluent and natural representation of the text [Ottensooser et al. 2012]. More specifically, we considered a natural language text which requires no background knowledge of layout or specific patterns. Also, our research focus on process understanding, while Ottensooser *et al.* focus on domain understanding through different representations. Nevertheless, our findings corroborates to Ottensooser *et al.* results which indicate there is no significant superiority between using graphical or textual notation for describing business processes.

6.2. Process Models to Textual Descriptions Generation

Our work also inspected related work where natural language techniques was used to achieve BPM relevant goals or areas where the creation of process models was the focus. The main challenge for generating text from process models is to adequately analyze the existing natural language fragments from the process model elements, and to organize the information from the process model in a sequential fashion. Some approaches provided initial insights for the construction of the framework.

The work presented by Leopold *et al.* describe an approach which automatically transforms BPMN process models into natural language texts combining different techniques from linguistics and graph decomposition [Leopold et al. 2012]. It is based on the NLG pipeline defined by Reiter and Dale [Ehud Reiter 1997]. The evaluation of the technique was based on a prototype implementation and involved a test set of 53 BPMN process models showing that natural language texts can be generated successfully. Following the same stream of research, Leopold *et al.* proposed a new approach which supports process model validation through natural language generation [Leopold et al. 2014]. Their work focus on using the generated texts for aiding the domain experts in the validation task, given the diverging skills set of domain experts and system analysts. Nevertheless, although the base generation technique has been introduced, as opposed to our work, both approaches do not support any other language besides English and do not provide an extensible framework able to be used to handle models designed using other languages (*e.g.*, Spanish, Dutch). Another drawback is its architecture, which is not detailed, making it difficult to be used or extended to fit specific needs that were not defined beforehand.

7. Conclusion

Process models are frequently used in various organizations for understanding, documenting, and visualizing tasks to be performed by enterprise employees. Through the approach of Natural Language Genaration (NLG), we enable non-technical users to understand process models disregarding the process model notation that was used for designing the models. Thus, the focus of this work is on the information being transmitted by text that represents process models.

We proposed a language-independent framework that automatically generates natural language texts from BPMN business process models. We presented details of the framework architecture and how the framework handles the NLG pipeline. The framework is able to handle new languages belonging to Romanian and Germanic sub-branches

of the Indo-European language family. Hence, our evaluation results cannot be automatically transferred to other language families, as for instance Asian languages. However, our approach is designed as a language independent solution, which can be theoretically applied to any language.

The framework analysis resulted in two main conclusions. First, the textual work instructions can be considered equivalent, in terms of knowledge representation, to process models within an acceptable threshold, since 74% of the participants claim that the equivalence between both knowledge representations vary from 100% to 68%. Second, our evaluation indicates the chosen textual format is good, since 86% of the participants claim the textual descriptions vary from excellent to good. These results are mainly due to the use of NLG techniques, like *Discourse Marker* insertion and *Referring expression* generation, which enhance text readability, as well as the use of bullets and indentation.

As future work, we suggest use the tool in a real business scenario. We further suggest adding new languages to the tool. For new languages, it is necessary to implement the specific interfaces defined in the *GeneralLanguageCommon* package (Figure 2). Suitable languages for this test would be, among others, German and Spanish.

References

Bajwa, I. S. and Choudhary, M. A. (2006). Natural language processing based automated system for uml diagrams generation. In *The 18th Saudi National Computer Conf. on computer science (NCC18). Riyadh, Saudi Arabia: The Saudi Computer Society (SCS)*.

Castro, L., Baião, F., and Guizzardi, G. (2011). A semantic oriented method for conceptual data modeling in ontouml based on linguistic concepts. In *Conceptual Modeling–ER 2011*, pages 486–494. Springer.

Curtis, B., Sheppard, S. B., Kruesi-Bailey, E., Bailey, J., and Boehm-Davis, D. A. (1989). Experimental evaluation of software documentation formats. *Journal of Systems and Software*, 9(2):167–207.

Dumas, M., La Rosa, M., Mendling, J., and Reijers, H. A. (2013). *Fundamentals of business process management*. Springer.

Ehud Reiter, R. D. (1997). Building applied natural language generation systems. *Natural Language Engineering 1*.

Friedrich, F., Mendling, J., and Puhlmann, F. (2011). Process model generation from natural language text. In *Advanced Information Systems Engineering*, pages 482–496. Springer.

Green, T. R., Petre, M., and Bellamy, R. (1991). Comprehensibility of visual and textual programs: A test of superlativism against the'match-mismatch'conjecture. *ESP*, 91(743):121–146.

Gruhn, V. and Laue, R. (2007). What business process modelers can learn from programmers. *Science of Computer Programming*, 65(1):4–13.

Haisjackl, C. and Zugal, S. (2014). Investigating differences between graphical and textual declarative process models. In *Advanced Information Systems Engineering Workshops*, pages 194–206. Springer.

Henver, A., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 28(1):75–105.

Kettinger, W. J., Teng, J. T., and Guha, S. (1997). Business process change: a study of methodologies, techniques, and tools. *MIS quarterly*, pages 55–80.

Larkin, J. H. and Simon, H. A. (1987). Why a diagram is (sometimes) worth ten thousand words. *Cognitive science*, 11(1):65–100.

Larman, C. (2005). Applying uml and patterns: An introduction to object-oriented analysis and design and iterative development.

Leão, F., Revoredo, K., and Baião, F. (2013). Learning well-founded ontologies through word sense disambiguation. In *Intelligent Systems (BRACIS), 2013 Brazilian Conference on*, pages 195–200. IEEE.

Leopold, H., Eid-Sabbagh, R.-H., Mendling, J., Azevedo, L. G., and Baião, F. A. (2013). Detection of naming convention violations in process models for different languages. *Decision Support Systems*, 56:310–325.

Leopold, H., Mendling, J., and Polyvyanyy, A. (2012). Generating natural language texts from business process models. In *Advanced Information Systems Engineering*, pages 64–79. Springer.

Leopold, H., Mendling, J., and Polyvyanyy, A. (2014). Supporting process model validation through natural language generation.

Mel'čuk, I. A. and Polguere, A. (1987). A formal lexicon in the meaning-text theory:(or how to do lexica with words). *Computational linguistics*, 13(3-4):261–275.

Mendling, J. (2009). Empirical studies in process model verification. In *Transactions on petri nets and other models of concurrency II*, pages 208–224. Springer.

Mendling, J., Moser, M., Neumann, G., Verbeek, H., van Dongen, B. F., and van der Aalst, W. M. (2006). *Faulty EPCs in the SAP reference model*. Springer.

Mendling, J., Reijers, H. A., and Cardoso, J. (2007). What makes process models understandable? In *Business Process Management*, pages 48–63. Springer.

Mendling, J., Reijers, H. A., and Recker, J. (2010). Activity labeling in process modeling: Empirical insights and recommendations. *Information Systems*, 35(4):467–482.

Mendling, J., Strembeck, M., and Recker, J. (2012). Factors of process model comprehension findings from a series of experiments. *Decision Support Systems*, 53(1):195–206.

Moher, T. G., Mak, D., Blumenthal, B., and Levanthal, L. (1993). Comparing the comprehensibility of textual and graphical programs. In *Empirical Studies of Programmers: Fifth Workshop*, pages 137–161. Ablex, Norwood, NJ.

Ottensooser, A., Fekete, A., Reijers, H. A., Mendling, J., and Menictas, C. (2012). Making sense of business process descriptions: An experimental comparison of graphical and textual notations. *Journal of Systems and Software*, 85(3):596–606.

Paas, F., Renkl, A., and Sweller, J. (2003). Cognitive load theory and instructional design: Recent developments. *Educational psychologist*, 38(1):1–4.

Polyvyanyy, A., Vanhatalo, J., and Völzer, H. (2011). Simplified computation and generalization of the refined process structure tree. In *Web Services and Formal Methods*, pages 25–41. Springer.

Pree, W. (1994). Meta patterns a means for capturing the essentials of reusable object-oriented design. In *Object-oriented programming*, pages 150–162. Springer.

Rodrigues, R., Azevedo, L., Revoredo, K., and Leopold, H. (2014). A tool to generate natural language text from business process models. In *CBSoft 2014 - Tool Session*.

Shneiderman, B., Mayer, R., McKay, D., and Heller, P. (1977). Experimental investigations of the utility of detailed flowcharts in programming. *Communications of the ACM*, 20(6):373–381.

Siau, K. (2004). Informational and computational equivalence in comparing information modeling methods. *Journal of Database Management (JDM)*, 15(1):73–86.

Winn, W. (1994). Contributions of perceptual and cognitive processes to the comprehension of graphics. *Advances in psychology*, 108:3–27.

Zugal, S., Pinggera, J., and Weber, B. (2011). Assessing process models with cognitive psychology. In *EMISA*, volume 190, pages 177–182.