BPMN Model and Text Instructions Automatic Synchronization

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

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

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

Keywords: BPM, BPMN, Natural Language Generation, Natural Language Processing.

Abstract: The proper representation of business processes is important for its execution and understanding. BPMN is the *de facto* standard notation for business process modeling. However, domain specialists, which are experts in the business, do not have necessarily the modeling skills to easily read a BPMN model. Natural language is easier to read for them. So, both model and text are necessary artifacts for a broad communication. However, unilateral manual editions of them may result in inconsistencies. This research proposes a framework for synchronizing BPMN model artifacts and its natural language text representation. It generates textual work instructions from the model, and it updates the original model if the textual instructions are edited. The framework was implemented using Java standard technology and evaluated through experiments. In the first experiment, we showed the knowledge represented by the textual work instructions and the correspondent process models are equivalent. Furthermore, in a second experiment, we showed our approach for maintaining the texts and models consistent performed satisfactory, where we verified the equivalence of the two artifacts.

1 INTRODUCTION

Many companies maintain both process models and textual work instructions to depict their processes (van der Molen, 2011). The use of both representations is needed to address specific audience. Usually, domain experts are not qualified for reading process models, due to lack of knowledge in the modeling notation, and they rely on the textual descriptions. On the other hand, modeling experts prefer using the model representation which are easier for them to read and understand. Companies face redundant effort for updating both process knowledge representation artifacts. This task is error-prone, and may bring inconsistencies.

The scientific problem addressed by this work is how to maintain both knowledge representation artifacts (texts and models) synchronized automatically, ensuring their consistency. We aim at enabling modelers to generate texts from models and domain experts to create or update formal models through textual descriptions writing and editions. As a result, the proposal leverages the information potential of already existing text documents. It preserves process verbalization techniques. It provides substantial savings, reducing manual efforts. It solves the inconsistency model-text problem. It enables a quicker realization of BPM-projects and their benefits.

This work proposes a *round-trip* framework for automatically maintaining the consistency of processes representations by reflecting to a process model modifications done to the text and vice-versa. The approach generates natural language text from a process model, and updates the process model from edited text. To perform these synchronizations, the framework creates correlations among process model's elements and the text. So, if one change one of the artifacts, it can use the framework to automatically update the other.

Using our approach domain specialists do not need to be trained in a specific business process modeling notation or in process modeling discipline. They can update a process model by editing the natural language text derived from the original process model. On the other hand, system analysts are relieved from the time-intensive task of writing and updating natural language texts that represents the models.

The remainder of this work is structured as follows. Section 2 presents the proposed `round-trip` framework and its implementation. Section 3 presents the evaluation. Section 4 presents the related works. Finally, Section 5 presents the conclusions, proposals of future work and existing framework's limitations.

Guerreiro Azevedo, L., de Almeida Rodrigues, R. and Revoredo, K.
BPMN Model and Text Instructions Automatic Synchronization.
DOI: 10.5220/0006809604840491
In *Proceedings of the 20th International Conference on Enterprise Information Systems (ICEIS 2018)*, pages 484-491
ISBN: 978-989-758-298-1
Copyright © 2019 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

2 THE ROUND-TRIP FRAMEWORK

This section presents the *round-trip framework*. Section 2.1 presents an overview of the framework and the scenarios it supports. Section 2.2 presents the trip to generate text from business process models, while Section 2.3 presents the trip to generate business process models from text instructions.

2.1 Framework Overview and Supported Scenarios

Figure 1 gives an abstract overview of the round-trip framework. One can starts from models or texts, top and bottom of the figure, respectively. The links between model's elements and text's sentences (middle of the figure) are created through pattern matching and stored for synchronization queries executed by Process Model to Natural Language (labeled as *P*) and Natural Language to Process Model (labeled as *N*) components. The pattern matching rules are defined as templates.

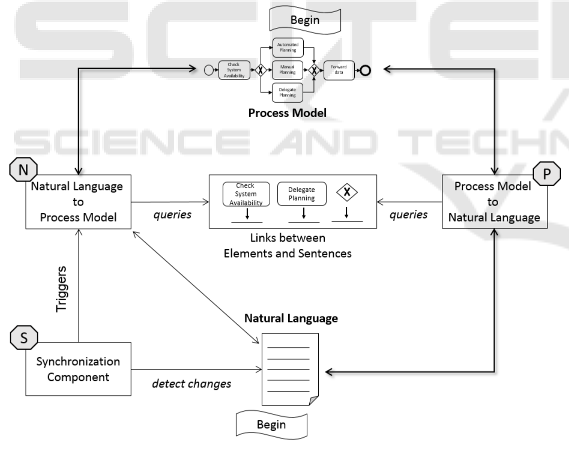
Begin
Automated Planning
Check System Availability
Manual Planning
Forward data
Delegate Planning
Process Model
N
Natural Language to Process Model
queries
Check System Availability
Delegate Planning
X
queries
Process Model to Natural Language
P
Links between Elements and Sentences
Triggers
Natural Language
S
Synchronization Component
detect changes
Begin

Figure 1: Abstract overview of the round-trip framework.

The framework fits the following scenarios:

- **Generate Text from Process Model**: The framework receives a model as input, triggers the Process Model to Natural Language component, which queries the linked elements, and generates the whole text as natural language sentences.
- **Generate Model from Process Textual Descriptions**: The framework receives a text as input, triggers the Natural Language to Process Model component, which queries the linked elements, and generates the whole model.
- **Update A Model from Edited Text**: The framework's Synchronization Component (labeled as *S*) detects changes in the text, and reflects them to the process model. This is done by triggering the Natural Language to Process Model component and, thus, generating an updated version of the model. This scenario only reflects changes, updating only the process model's elements that were altered.
- **Update a Text from Edited Model**: The text update is handle as a new text generation scenario, *i.e.*, the whole model is submitted as input to the Process Model to Natural Language component and the whole text is regenerated. The decision to generate the whole text again instead of updating only sentences which had changes was based on cost-benefit analysis, considering the performance gain and development effort.
- **Round-Trip**: This scenario is represented by a generation cycle. For illustration purpose, consider the process model as starting point. The model is submitted as the framework's input, and the text is generated. Then, the text is manually changed and the original process model is automatically updated. In the same cycle, the updated model is submitted again to assert whether it matches the text description. At this point, the user can go in both directions, generate text from model or model from text.

The NLG and NLP core components are composed by several ready-to-use building blocks (*Frozen spots*) and define interfaces which must be implemented to support specific languages (*Hot spots*). Interfaces provide the flexibility to satisfy specific needs. For instance, regarding NLG, each hot spot can be implemented for a specific language (*e.g.*, German and Spanish) without the need to change any component. Regarding NLP, the hot spots can be implemented to override the default behaviour or to offer support to different text formats (*i.e.*, text patterns). The architecture's frozen spots are represented by classes, while the hot spots are represented by interfaces (Pree, 1994).

Figure 2 presents a package diagram of the hot spots implementations developed for Portuguese and English. They are suffixed as *Realizer* since they realize the implementations of `GeneralLanguageCommon` package interfaces. Each language has its own specific implementation (*e.g.*, `PortugueseLabelHelper` class implements `ILabelHelper`). So, `PortugueseRealizer` and `EnglishRealizer` classes implement hot spots and use frozen spots to accomplish necessary tasks.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

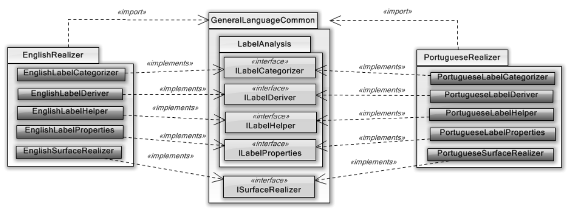

Figure 2: Implementation of the hot spots defined by the architecture.

2.2 Model to Text: Natural Language Generation from BPMN Process Model

This section describes the component Process Model to Natural Language, *i.e.*, the NLG component.

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 in a sequential fashion. The *Model to Text* component of the framework was implemented following the Three-Step NLG pipeline proposed by Reiter and Dale (Ehud Reiter, 1997), which are:

- *Text Planning*: Determines the information to be communicated in the text, and specifies the order this information should be conveyed.
- *Sentence Planning*: Chooses specific words to express the information determined in the previous step. It may aggregate messages and include pronouns in the text to obtain smoothly text.
- *Sentence Realization*: Transforms the messages into grammatically correct sentences.

The pipeline was instantiated as presented in Figure 3 and detailed as follows:

- *Text Planing*
  - *Linguistic Information Extraction*: Decomposes process model element's labels using a linguistic label analysis technique (Leopold et al., 2013). For instance, it decomposes the activity label *Inform customer about problem* into action (*inform*), business object (*customer*), and addition (*about problem*).
  - *Annotated RPST Generation*: Creates Refined Process Structure Tree (RPST) (Vanhatalo et al., 2009)) from the process model.
  - *Text Structuring*: Annotates the RPST tree's nodes with the linguistic information obtained in the first step.
- *Sentence Planning*
  - *DSynT-Message Generation*: Maps the annotated RPST elements to a list of intermediate messages. Each sentence is stored as a Deep-Syntactic Tree (DSynT) (Mel'čuk and Polguere, 1987).
  - *Message Refinement*: Aggregates messages, generates referring expressions (*e.g.*, replace the role *analyst* to *he*), and inserts discourse marker (*e.g.*, *afterwards* or *subsequently*) if needed. It performs these refinements usually on long sequences of tasks.
- *Realization*
  - *Surface Realization*: Transforms the intermediate messages into grammatically correct sentences. This is accomplished by systematically mapping the generated DSynT to the corresponding grammatical entities.

2.3 Text to Model: BPMN Process Model Generation from Natural Language Texts

This section describes the component Natural Language to Process Model, *i.e.*, the NLP component. It uses NLP techniques to extract and parse texts to identify BPMN process model elements and create/update the model. It was implemented as presented in Figure 4 and detailed as follows:

- *Text Planning*
  - *Text Pattern Extraction*: Infers the textual pattern from the text received as input. It validates if the text received as input follows the supported text patterns, thus saving both user's time and system processing resources.
  - *Natural Language Processing*: Analyzes the text's semantic and extracts relevant linguistic information. It employs NLP techniques to: remove stop words (*e.g.*, articles and discourse markers, like *Then*, *Afterwards*); recognize patterns (*e.g.*, end of line); split the text into sentences; split the sentences into words array and classify words according to their respective syntactic rule, *i.e.*, it executes Part-Of-Speech Tagging (POS), which consists of tagging each word with its respective syntactic role (*e.g.*, plural noun, adverb).
  - *Text Cleaning*: Replaces referring expressions and sentences aggregations (*e.g.*, replaces *he* by the role *analyst*).
- *Sentece Text Planning*

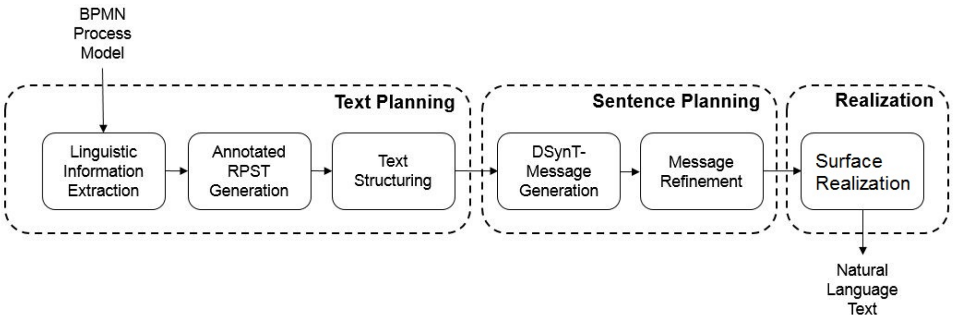

Figure 3: Extended pipeline approach for NLG.

- *Sentence Text Generation and Structuring*: Generates *SentenceText* objects for the original *RealizedText* while preserving the description structure, *e.g.*, generates activities in the correct sequence order and correlation among them.

• *Process Moderl Realization*

- *Model Elements Generation*: Generates the model elements from the *SentenceText* objects. The support of multiple text structures and formats is this step challenge.
- *Process Model Structuring*: Structures the generated elements according to the order which they appear within the text and also respecting sentences correlations. It takes advantage of the link between the *SentenceText* and the associated model elements, and it benefits from the initial sentences structuring. Each *SentenceText* is mapped only to one model element, and it is just needed to traverse through the sentences tree nodes, fetch its associated model element and create the *ProcessModel* object. With the *ProcessModel* object (which is not notation specific), it is possible to translate it to any specific notation and print out the whole process model to the desired output (*e.g.*, JSON, XML, BPMN, etc), which is done by the next and final NLP's pipeline component.
- *BPMN Model Generation*: Generates a business process model in a BPM notation. In this work implementation, the used notation was BPMN. It generates a JSON file that represents the process model as a machine artifact. The JSON file can then be used in a BPMN graphic modeler tool to dispose all the elements visually. In particular, Signavio[1] online platform was used to read the JSON file and output the graphic process model to the user.

[1]available at www.signavio.com

3 EVALUATION

This section presents the evaluation of the techniques developed within the language-independent framework.

The first experiment's objective was *Assess whether the knowledge represented by the generated process description can be considered equivalent to the process model*. It was guided by the following research question (RQ1): *"Is the knowledge represented by the natural language text, generated by the framework, equivalent to the process model?"*. Thus, the focus were in determining how many answers[2] were within the equivalence range varying between 100 and 68%, how many were between 67 and 34%, and, finally, how many were between 33 and 0%. The expectation was the number of answers between 100 and 68% be higher than the sum of the other two groups (participants who rate equivalence between 67 and 0%).

Figure 5 depicts the overall evaluation for this experiment. As can be observed, 342 answers were within the equivalence group ranging from 100% to 68%, which can be read as *74% of the participants claim that the equivalence between both knowledge representations vary from 100% to 68%*[3]. 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.*

[2]The questionnaire is available at https://docs.google.com/forms/d/1zvuuxojWUVWnyRpMsao-F1Yjygs-Dm2ebfrMjjjJqx4/viewform (in Portuguese)

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

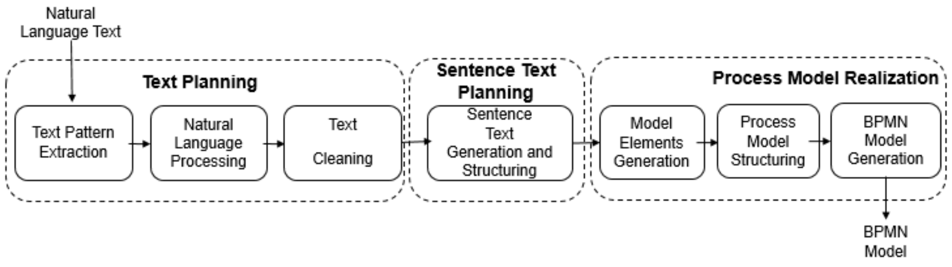
Natural Language Text
Text Planning
Text Pattern Extraction
Natural Language Processing
Text Cleaning
Sentence Text Planning
Sentence Text Generation and Structuring
Process Model Realization
Model Elements Generation
Process Model Structuring
BPMN Model Generation
BPMN Model

Figure 4: Text to model pipeline.

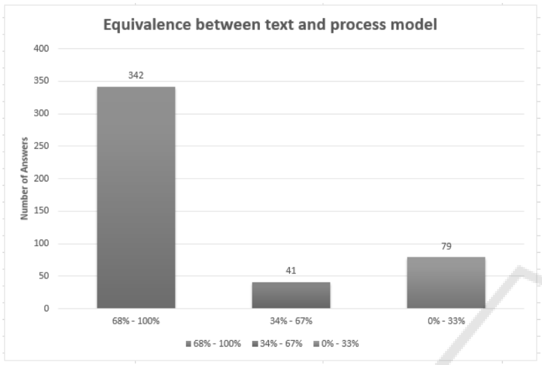

Figure 5: Subject's answers distribution among equivalence intervals.

The second experiment's objective was *Assess whether the knowledge represented by the automatically updated version of the process model, after been synchronized by text-based changes, represents the same knowledge as the manually updated textual description*. It was guided by the following research question (RQ2): *"Is the knowledge represented by the manually updated text equivalent to the automatically updated process model?"*. Thus, the focus was to determine how many answers were within the range varying from *Strongly Disagree* to *Strongly Agree*. The expectation was the number of answers[4] between *Strongly Agree* and *Slightly Agree* be higher than the sum of the other groups (subjects who rate accordance with the synchronization strategy between *Neutral* and *Strongly Disagree*). Thus, to enable a better reading of the results, the answers for options *Strongly Agree* and *Slightly Agree* were grouped into one group. All the remaining answers were grouped into a second group.

Figure 6 depicts the overall evaluation for this question. As can be observed, 160 answers were within the equivalence group ranging from *Strongly Agree* and *Slightly Agree*, which can be read as *78% of the subjects*[5] *claims the knowledge represented by the manually updated text is equivalent to the automatically updated process model*. Based on this, the answer for RQ2 is: *The knowledge represented by the manually updated text can be considered equivalent to the automatically updated process model.*

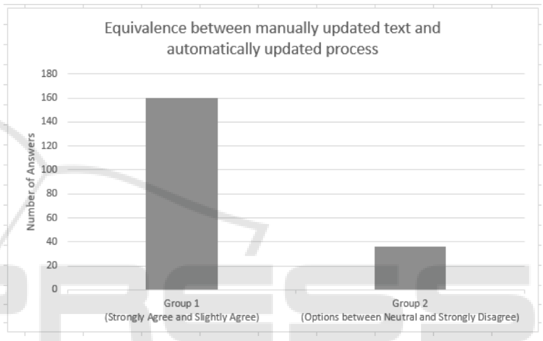

Figure 6: Subject's answers distribution among the available accordance options (grouped into two groups).

4 RELATED WORK

Several related works proposed similar approaches to our framework. They can be grouped into three (3) specific topics: (i) Text to model generation; (ii) Model to textual description generation; and, (iii) business process understandability.

4.1 Text to Model and Model to Text Generation

We inspected related works where natural language techniques was used to achieve BPM relevant goals or areas where the creation of process models was the focus. These approaches provided initial insights for the construction of the NLG Core module architecture). They are presented in Table 1 along with comparisons with our work.

[4]The questionnaire is available at http://goo.gl/forms/JacmFbV3yTpPEfib2 (in Portuguese)

[5]Each subject contributed with seven (14) answers. Thus, if we divide the total number of answers by fourteen (160/14), it gives us the average number of subjects that choose the same answer (11 subjects)

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

Table 1: Papers' Comparison.

Paper and Objective	Description	Comparision
Leopold *et al.* (Leopold et al., 2012): Generate natural language texts from BPMN models.	Approach that automatically transforms BPMN process models into natural language texts. It is based on Reiter and Dale's NLG pipeline (Ehud Reiter, 1997). Moreover, Leopold *et al.* proposed a new approach that validates process model through natural language generation (Leopold et al., 2014).	Our framework is capable of parsing process models and generating textual process descriptions from these models like Leopold *et al.* approach. However, our approach is capable of generating the model from the text and is not language specific.
Gonçalves *et al.* (de AR Goncalves et al., 2009): Generate conceptual models from natural language texts.	A method for deriving conceptual models from text focusing on the derivation of models from group stories. They provide a prototype which handles Portuguese texts.	As opposed to our work, theirs offer no support to any other language, it parses only story like texts, considered a limited set of BPMN elements and provides a simple evaluation. Furthermore, a couple of their exhibits show that syntactical problems occur in some cases.
Ghose *et al.* (Ghose et al., 2007): Generate UML compliant process models from natural language texts.	Toolkit that uses a syntax parser to identify verb-object phrases in the given text and also scans the text for textual patterns.	The results are BPMN model snippets rather than a fully connected model.
Kop and Mayr (Kop et al., 2005): Generate models from natural language texts.	Proposed a procedure called KCPM (Klagenfurt Conceptual Predesign Model) and developed a corresponding tool. It parses textual input in German and fills instances of a generic meta-model, the KCPM. Using the information stored in this meta-model, an UML activity diagram and a UML class diagram can be created.	As opposed to our work, the transformation from natural language input to the aforementioned meta-model is not a fully automated process and supports only German written texts.
Sinha *et al.* (Sinha and Paradkar, 2010): Generate process model from natural language text.	They employ a linguistic analysis engine based on the UIMA Framework, which enables the constructions of linguistic analysis systems by combining different blocks into a pipeline.	Their work does not contain a full text and model example, which does not allow a comparison with ours. Besides, it is a text type specific since it uses only use-case descriptions.
Wang *et al.* (Wang et al., 2009): Generate BPMN model from structured use cases.	Describe a procedure which creates a BPMN diagram, given that data items, tasks, resources (actors) and constraints are identified in an input text document.	Compared to Wang *et al.* work, our approach does not require user-interaction during the parsing execution. Ours supports 15 different BPMN elements, and does not impose any restriction regarding the number of gateways outgoing arcs. They also do not present any process model they use.
Friedrich *et al.* (Friedrich et al., 2011): Generate model from natural language text.	Proposes an automatic procedure for generating BPMN models from natural language text composed by an anaphora resolution component. The evaluation count with more than 45 models from both, academy and industry.	Their work generates a conceptual model from natural language text and does not generate natural language texts from the models. It is capable of parsing only English texts and they do not present possibilities to extend or adapt the technique to fit specific needs.

4.2 Business Process Understandability

The field of process model understandability is discussed from different perspectives.

Mendling *et al.* show the number of arcs has an important effect on the overall model understandability (Mendling et al., 2007). Mendling *et al.* demonstrates the impact of natural language in the activity labels for model comprehension (Mendling et al., 2010). Zugal *et al.* investigated how far the cognitive inference process affects the model understanding (Zugal et al., 2011). Leopold *et al.* build on these insights trying to lower the overall burden of process model comprehension (Leopold et al., 2014).

Prior 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. In a previous work, we run an experiment on process model understandability using textual work instructions and BPMN Models. The results were instructions or process models do not influence process understandability for non-expert users but do influence experienced users (Rodrigues et al., 2015).

Haisjackl and Zugal compared declarative process models against a text based notation using subjects with experience in modeling declarative process (Haisjackl and Zugal, 2014). Different from their work, ours used imperative process models, involved subjects with experience in process modeling varying from none to expert, and presented a natural language text simulating a human description of the process.

While the experiment described by Ottensooser *et al.* compares model understanding with written use cases (using the Cockburn format), our framework is based on a more fluent and natural representation of the text (Ottensooser et al., 2012). More specifically, it was 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 work's findings corroborates to Ottensooser's results which indicate there is no significant superiority between using graphical or textual notation for describing business process.

5 CONCLUSION

This work proposed a round-trip framework capable of automatically generating natural language text from process models and updating these models from text editions. It solves the problem of maintaining text and model elements representation synchronized, saves time and effort of analysts to manually write text descriptions, enables domain experts to edit formal process models without the efforts of learning a business process modeling language.

The round-trip framework combines existing tools from graph decomposition, natural language processing and generation in an innovative way. The framework provides the foundations for integrating textual and model-based information, meeting the challenge of processing both textual process descriptions and models.

From a practical perspective, the framework helps organizations to simplify business process management since: (i) Creates automatically business process instructions in natural language text; (ii) Allows update the process model from text editions; (iii) Reduces efforts required by a business analyst which can use the text to understand the processes.

The first experiment's concluded the textual work instructions can be considered equivalent, in terms of knowledge representation, to process models within an acceptable threshold. 74% of the subjects claims the equivalence between both knowledge representations vary from 100% to 68%, and 86% of the subjects claims the textual descriptions vary from excellent to good. This result is aligned with what we expected due to the use of NLG techniques like Discourse Marker insertion and Referring expression generation, which are capable of enhancing the text and improving its readability. The second experiment concluded the knowledge represented by the manually updated text can be considered equivalent to the automatically updated process model after the synchronization within an acceptable threshold. 78% of the subjects claims knowledge represented by the manually updated text is equivalent to the automatically updated process model.

Despite these encouraging results, the framework is able to read process descriptions consisting of full sentences. Another prerequisite is the text be grammatically correct, does not contain questions, and has little process irrelevant information. Also, Word Sense Disambiguation and Named Entity Recognition was not within the scope. Another issue is the test data set, which comprised thirty text-models. It is a relatively small dataset. Therefore, this research's test results are not fully generalizable.

Some future work are: evolve the framework to a "human in the loop" approach through the interaction with domain experts to provide missing information interactively, and further relive the business analyst from interviewing tasks; support specification or overriding the text patterns; analyze and evolve it to process large amounts of textual information, cover other BPMN elements, include charts and other data models, and add new languages.

REFERENCES

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.

de AR Goncalves, J. C., Santoro, F. M., and Baiao, F. A. (2009). Business process mining from group stories. In *Computer Supported Cooperative Work in Design, 2009. CSCWD 2009. 13th International Conference on*, pages 161–166. IEEE.

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.

Ghose, A., Koliadis, G., and Chueng, A. (2007). Rapid business process discovery (r-bpd). In *Conceptual Modeling-ER 2007*, pages 391–406. Springer.

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.

Kop, C., Vöhringer, J., Hölbling, M., Horn, T., Mayr, H. C., and Irrasch, C. (2005). Tool supported extraction of behavior models. In *ISTA*, volume 63, pages 114–123.

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., 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.

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.

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. D. A., Barros, M. D. O., Revoredo, K., Azevedo, L. G., and Leopold, H. (2015). An experiment on process model understandability using textual work instructions and bpmn models. In *Software Engineering (SBES), 2015 29th Brazilian Symposium on*, pages 41–50.

Sinha, A. and Paradkar, A. (2010). Use cases to process specifications in business process modeling notation. In *Web Services (ICWS), 2010 IEEE International Conference on*, pages 473–480. IEEE.

van der Molen, T. (2011). Maintaining consistency between business process diagrams and textual documentation using the epsilon model management platform.

Vanhatalo, J., Völzer, H., and Koehler, J. (2009). The refined process structure tree. *Data & Knowledge Engineering*, 68(9):793–818.

Wang, H. J., Zhao, J. L., and Zhang, L.-J. (2009). Policy-driven process mapping (pdpm): Discovering process models from business policies. *Decision Support Systems*, 48(1):267–281.

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

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS