A Concept for Generating Business Process Models from Natural Language Description

Krzysztof Honkisz, Krzysztof Kluza(✉), and Piotr Wiśniewski

AGH University of Science and Technology,
al. A. Mickiewicza 30, 30-059 Krakow, Poland
honkisz@student.agh.edu.pl, {kluza,wpiotr}@agh.edu.pl

Abstract. Manual extraction of business process models from technical documentation is a time-consuming task. Several approaches to generating such process models have been proposed. We present a proposal of a new method for extracting business process from natural language text through intermediate process model using the spreadsheet-based representation. Such intermediate model is transformed into a valid BPMN process model. Our method is enhanced with semantic analysis of the text, allows for quick check of the transformation result and manual correction during this process. As the obtained BPMN model is structured, it is easier to check its correctness.

1 Introduction

Business process management plays an important part in modern corporation and enterprise management. Business process models can be used as a documentation for work-flow implementation, partial automation or optimization of process. One of the most popular standards providing graphical representation of processes is Business Process Model and Notation (BPMN) [1].

Usually, some knowledge about the processes and work-flows in the company exists either in the form of human knowledge or as a textual documentation. Manual extraction of a process model from technical documentation (textual description) is a time-consuming task. Since every enterprise must constantly improve its services, their process models must be frequently updated. Moreover, manually designed models can be different, depending on the designer's experience and knowledge.

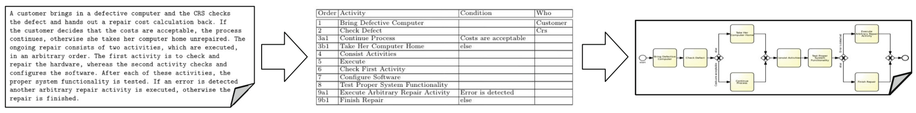
To solve this problem, several approaches of automatic business process models generation have been proposed in recent years [2–4] An effective way of machine-aided transformation from a semi-formal document into a process model can provide a significant savings in time, additionally making maintenance of formal process models and documentation easier. Furthermore, an automatic tool

The paper is supported by the AGH UST research grant.

for model extraction can be very useful for people, who do not have the sufficient knowledge and expertise in the process modelling field.

In this paper, we present a concept of a new method for extracting business process from natural language text through intermediate process model based on the spreadsheet representation. The overview of the approach is presented in Fig. 1. The intermediate model uses our structured spreadsheet-based representation for describing business process models. The method of obtaining this model is based on the syntactic analysis of a given natural text and extracting Subject-Verb-Object construct (SVO), which can be later transformed into process activities. Our method is enhanced with semantic analysis of the text, which allows to filter out unnecessary SVO constructs and transform them into valid activities names. Then, the spreadsheet representation is transformed into a BPMN process model.

The method was implemented using our `bpmn_python` library and tested on a set of natural language business process descriptions, gathered from different BPMN tutorials and academic sources. Thanks to this intermediate step, our method allows for quick check of the transformation result and manual correction of the spreadsheet, as well as the obtained result is a structured BPMN model what can help in correctness checking and verification.



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

Order	Activity	Condition	Who
1	Bring Defective Computer		Customer
2	Check Defect		Crs
3a1	Continue Process	Costs are acceptable	
3b1	Take Her Computer Home	else	
4	Consist Activities		
5	Execute		
6	Check First Activity		
7	Configure Software		
8	Test Proper System Functionality		
9a1	Execute Arbitrary Repair Activity	Error is detected	
9b1	Finish Repair	else	

Fig. 1. Overview of the presented approach

The rest of this paper is organized as follows: Sect. 2 presents the related works in the field of process model extraction from textual description. In Sect. 3, we give an overview of our solution. Section 4 shows a case study example. The paper is summarized in Sect. 5.

2 Related Works

Terins and Thaler performed an analysis of current state-of-the-art in the field of mining process models from natural language text [3]. They presented several approaches, some of which worked with some form of structured text (use cases, group stories), some with natural language description.

Ghose, Koliadis and Chueng [5] proposed an approach for discovering a process model from text artefacts, which are described as *documents such as memos, manuals, requirements documents, design documents, mission/vision statements, meeting minutes etc.* An extraction is performed by text pattern search (for example if/then pair, which indicates a conditional flow) and POS (*Part-Of-Speech*) tagging combined with shallow parsing, which produces a syntax tree.

This approach allows to discover parts of a larger model (called *proto-models* by authors), rather than complete and sound model. Generated *proto-models* can be compared in order to find similarities and remove redundant parts.

Goncalves et al. [6] presented a technique of obtaining process models from group stories. In the first phase, a text is tokenized in order to select words which can be useful for work-flow generation. Next, a POS tagging is performed. Finally, relevant entities are identified using a set of predefined patterns. The produced BPMN model is not necessarily complete and sound – it is assumed, that it will be later improved by process designer and team members, who created the group stories.

Friedrich et al. [7] proposed an advanced approach, which uses a textual description of model. Such a description must follow some requirements – a text cannot contain any questions and the described execution of a process must be sequential (any non-sequential jumps must be explicitly made). In the first step, a syntactical analysis (called *Sentence Level Analysis* in the article) is performed, using Stanford NLP tools. Next, the semantic analysis (*Text Level Analysis*), using WordNet and FrameNet databases, allows to identify relevant entities. Finally, the process model is generated (*Process Model Generation* phase). The generated output is a sound and complete BPMN model, enriched with many additional elements (such as lanes, data sources), thanks to the rich text analysis.

Several other methodologies for transforming natural language text into formalised models were proposed. As there are various notations for representing process models [8], the related approach not always use the BPMN model. Yue, Briand and Labiche [9] presented an automated approach to transform use case descriptions to UML Activity diagrams. This methodology requires that the use case descriptions has to follow some restriction rules. These rules can be classified into two groups – the first group specifies constraints on the use of natural language, the second are requirement on the use of specific keywords to indicate the existence of control structures. In addition, the use case description explicitly lists all of the flows in the process (main and alternative) and each flow is a step-by-step description of a process.

Another approach in the field of generating formal models form natural language specification, proposed by Njonko and Abed [10], uses Semantics of Business Vocabulary and Business Rules (SBVR) as an intermediate layer for this transformation approach. It is suggested that using formalised model as an intermediate layer (in this case SBVR), it is possible to easily extend this approach for multiple models. The article presents an example of transformation from natural language business description into SQL executable query, which produces a database table that corresponds to business requirements.

As SBVR is in fact a textual description, but using controlled structured language, in our previous approach, we also developed the approach which extracts process models from the SBVR description [4]. A structured text description can be generated by searching natural language documents for keywords related to business process models. Ferreira, Thom and Fantianto [2] propose a method which is based on a syntactic analysis of natural text such as forms, e-mail messages and reports in order to generate a tagged document. This step is followed by a logical analysis that uses a set of predefined rules to identify flow objects and swimlanes of the model which is a basis for process-oriented structures.

Identification of business process objects in natural language texts may also point out elements missing in the textual description. Therefore, preprocessing the process specification may facilitate the modeling phase. According to the survey [11], over 60% of experienced BPMN modelers find creating a business process model easier if a rule-mapped text is used as a specification, in comparison with natural language descriptions.

Structured forms generated as a result of natural language processing may suffer from various inconsistencies when using textual specifications from different sources, such as user manuals, instructions and standards. These documents may provide incomplete or contradictory information which in the extreme case may lead to the incapability of generating a correct model. To overcome this limitation, an idea of semantic unification was presented in [12]. The feature structures identified in the text can be mapped to attribute-value matrix objects which are then unified to a single description.

A comparative analysis of selected NLP-based process modeling approaches was performed by Riefer, Ternis and Thaler [3]. The authors compare the approaches based on three common pillars, namely: textual input, text analysis and model generation. The obtained results show that although most of the existing approaches do not generate complete BPMN models, they provide a firm basis for further process modeling.

Table 1 present a comparison of the mentioned approaches regarding their input and output data and the ability to generate BPMN diagrams.

Table 1. Comparison to the existing approaches

Approach	Input	Output	Method	BPMN support
Yue et al. [9]	Use case descriptions	UML activity diagram	Rule-based	○
Njoko and Abed [10]	NL specification	SQL query	SBVR	○
Ghose et al. [5]	Text documents	Proto-models	Text pattern search	◐
Ferreira et al. [2]	Natural text	Process structure	Rule-based	◐
Sokolov et al. [12]	Natural text	Unified description	SVM structures	◐
Kluza and Honkisz [4]	SBVR description	Process model	Rule-based	●
Goncalves et al. [6]	Group stories	BPMN model	POS tagging	●
Friedrich et al. [7]	Sequential description	BPMN model	Semantic analysis	●
Our approach	Natural text	BPMN model	Spreadsheet-based	●

3 Algorithm for Generating Business Process Models

This section describes the proposed approach to business process model generation from natural language description. This approach can be divided into the following steps:

1. Participants extraction – in this step, a sentence from a given description is analysed and the information about possible participants (people, systems or organizations which performs the tasks) in process are extracted,
2. Subject-verb-object constructs extraction – a sentence from given description is analysed in search of basic SVO constructs, which later will be used to create appropriate BPMN elements,
3. Gateway keywords search – a process description is analysed in search of the keywords that signalizes the presence of conditional (exclusive or inclusive) and parallel gateways,
4. Intermediate process model generation – an intermediate model in the form of spreadsheet-based representation is created from the acquired data,
5. BPMN diagram generation – a BPMN diagram is generated from the intermediate process model.

Figure 2 shows the overview of proposed approach. The generated intermediate model is parsed to BPMN diagram, using functionality provided by `bpmn_python` library. The prototypical tool implemented for the purpose of this paper generates both spreadsheet-based intermediate model and BPMN diagram, which makes the result analysis easier.

3.1 Participants Extraction

In the first step, each sentence of the description is analysed in search of words representing participants. This process is divided into three parts: (1) the sentence is analysed in search of specific dependency relations, namely nominal subject and nominal subject passive; (2) the sentence is searched for conjunction dependencies because the participant might be labelled as an object of the phrase; (3) simple semantic analysis is used to decide, whether the extracted words can be used as participants of process. The extracted word is added to output as a participant if it fulfills such conditions as the word is a pronoun or relative pronoun, one of its hypernyms belongs to the specified list of admissible hypernym keywords like `person` or `organization` or the word is a special keyword like ATM, CRM, etc.

A full name of the participant is extracted from its syntax sub-tree, provided that a given token from sub-tree is labelled with a correct dependency (see an example in Table 2 based on the syntax tree from Fig. 3.

3.2 Subject-Verb-Object Extraction

After extracting the participants from the sentence, syntactic analysis in search of SVO (subject-verb-object) constructs is performed. These construct are used to generate intermediate process model.

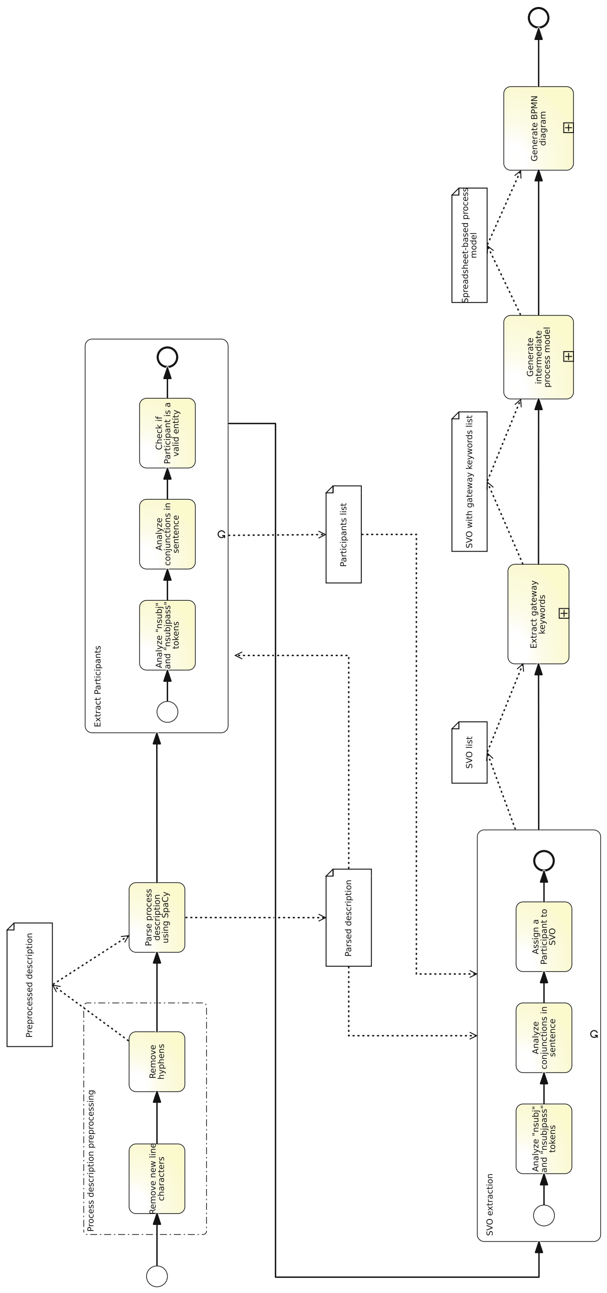

Process description preprocessing
Remove new line characters
Remove hyphens
Preprocessed description
Parse process description using SpaCy
Parsed description
Extract Participants
Analyze "nsubj" and "nsubjpass" tokens
Analyze conjunctions in sentence
Check if Participant is a valid entity
Participants list
SVO extraction
Analyze "nsubj" and "nsubjpass" tokens
Analyze conjunctions in sentence
Assign a Participant to SVO
SVO list
Extract gateway keywords
SVO with gateway keywords list
Generate intermediate process model
Spreadsheet-based process model
Generate BPMN diagram

Fig. 2. BPMN diagram with overview of proposed approach.

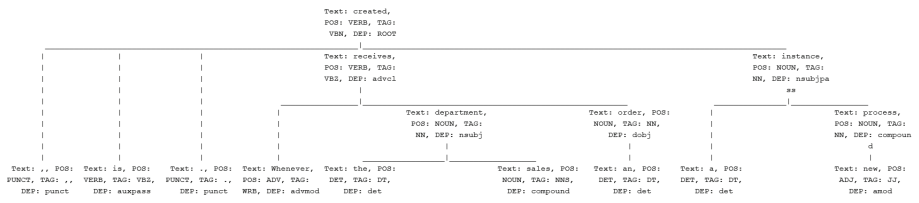

Text: created, POS: VERB, TAG: VBN, DEP: ROOT
Text: receives, POS: VERB, TAG: VBZ, DEP: advcl
Text: instance, POS: NOUN, TAG: NN, DEP: nsubjpass
Text: department, POS: NOUN, TAG: NN, DEP: nsubj
Text: order, POS: NOUN, TAG: NN, DEP: dobj
Text: process, POS: NOUN, TAG: NN, DEP: compound
Text: ,, POS: PUNCT, TAG: ,, DEP: punct
Text: is, POS: VERB, TAG: VBZ, DEP: auxpass
Text: ., POS: PUNCT, TAG: ., DEP: punct
Text: Whenever, POS: ADV, TAG: WRB, DEP: advmod
Text: the, POS: DET, TAG: DT, DEP: det
Text: sales, POS: NOUN, TAG: NNS, DEP: compound
Text: an, POS: DET, TAG: DT, DEP: det
Text: a, POS: DET, TAG: DT, DEP: det
Text: new, POS: ADJ, TAG: JJ, DEP: amod

Fig. 3. A syntax tree for the simple phrase *"Whenever the sales department receives an order, a new process instance is created."*

Table 2. Spreadsheet-based description generated from the sentence in Fig. 3

Order	Activity	Condition	Who
1	Receive order		Sales department
2	Create process instance		

First, the sentence is searched for nominal subject and nominal subject passive dependencies. For every word found, a new SVO construct is added to the output. In the case of words with nominal subject dependency, the subject is created from the extracted word, its predecessor in the syntax tree acts as a verb and the object is extracted from the subject's ancestors in syntax tree.

If the appropriate token is found, it is added as an object to SVO, otherwise the object part of SVO construct is omitted. In case of tokens with nominal subject passive dependency, the object is omitted. For example, in the sentence: *"Purchase is registered"*, the word *"Purchase"* is tagged as *"nsubjpass"* and no object is present.

Similarly to the participants extraction, the SVO extraction also analyses the existence of conjunction in sentences. This approach helps to deal with the sentences like: *"If the storehouse has successfully reserved or backordered every item"*. In this case, by conjunction analysis it is possible to extract construct *"backordered every item"*, which is conjoined by word *"backordered"*.

An example of SVO extraction is shown in Table 3.

Table 3. Spreadsheet-based description generated from the sentence in Fig. 4

Order	Activity	Condition	Who
1	Ship bicycle		Sales department
2	Finish process instance		Sales department

3.3 Gateway Keywords Search

After extracting the participants and subject-verb-object constructs, the description is analysed once more, in order to find keywords indicating the existence of possible gateway. This function searches for three different types of keywords: conditional, parallel and default flow. The first type can be later translated either into an exclusive or inclusive gateway, second – into a parallel one. Third type might be used as a default flow of conditional gateway, provided that the correspondence with a conditional keyword will be found during model generation. If no correspondence is found, the SVO will be treated as a simple activity.

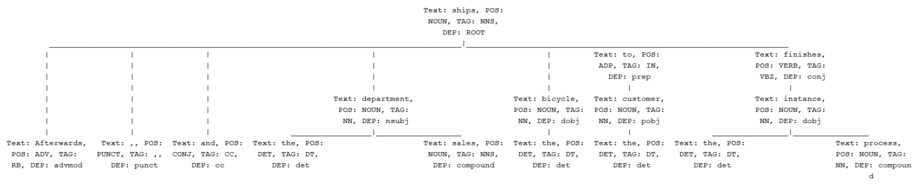

Text: ships, POS: NOUN, TAG: NNS, DEP: ROOT
Text: to, POS: ADP, TAG: IN, DEP: prep
Text: finishes, POS: VERB, TAG: VBZ, DEP: conj
Text: department, POS: NOUN, TAG: NN, DEP: nsubj
Text: bicycle, POS: NOUN, TAG: NN, DEP: dobj
Text: customer, POS: NOUN, TAG: NN, DEP: pobj
Text: instance, POS: NOUN, TAG: NN, DEP: dobj
Text: Afterwards, POS: ADV, TAG: RB, DEP: advmod
Text: ,, POS: PUNCT, TAG: ,, DEP: punct
Text: and, POS: CONJ, TAG: CC, DEP: cc
Text: the, POS: DET, TAG: DT, DEP: det
Text: sales, POS: NOUN, TAG: NNS, DEP: compound
Text: the, POS: DET, TAG: DT, DEP: det
Text: the, POS: DET, TAG: DT, DEP: det
Text: the, POS: DET, TAG: DT, DEP: det
Text: process, POS: NOUN, TAG: NN, DEP: compound

Fig. 4. Syntax tree for simple phrase *"Afterwards, the sales department ships the bicycle to the customer and finishes the process instance"*

An example of conditional flow extracted from the process description is shown in Table 4. Because the keyword associated with the default flow was found and an activity with condition *else* was added, an XOR gateway was added to the diagram (Fig. 5).

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

Text 1.1: Fragment of a process description with extractable conditional gateway

Table 4. Spreadsheet-based description generated from a text description shown in Text 1.1

Order	Activity	Condition	Who
1	Bring defective computer		Customer
2	Check defect		Crs
3a1	Continue process	Costs are acceptable	
3b1	Take her computer home	else	

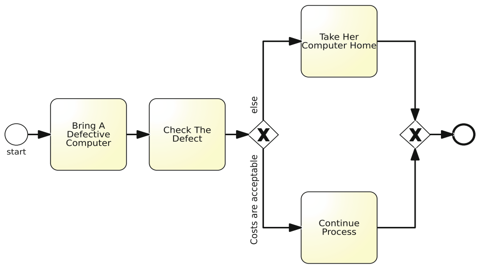


Fig. 5. BPMN diagram with an XOR gateway, generated from the intermediate process model shown in Table 4

3.4 Intermediate Model Description

The prototype implementation uses a spreadsheet-based process description [13], which employs a CSV (Comma-Separated Values) file format to represent a business process model. A business process is described by a spreadsheet table. Each row represents a single phase, which can be translated into a BPMN task or sub-process. Columns represent the properties of each phase[1], such as:

1. *Order* – the number of the corresponding phase (with the suitable suffixes for parallel or excluding tasks, and nested gateways).
2. *Activity* – the name of the performed action. There is a special case – `goto X` which signalizes a skipped part of the process or a loop.
3. *Condition* – the condition which has to be fulfilled in order to perform the task. This property is used to implement the exclusive and inclusive gateway.
4. *Who* – the name of participant (person, system or department) responsible for executing this phase.

The spreadsheet-based process description supports only basic BPMN elements. However, the subset of supported BPMN elements covers the most commonly used elements of BPMN diagram [14].

Based on the results from the previous phases, our tool iterates over a list of extracted SVO. The process starts with a start event, and then for each construct, the appropriate action should be performed:

– If the SVO is labelled with a conditional gateway keyword, it is added to the intermediate model as a part of conditional gateway. The property *Condition* is filled with a full name of the condition SVO. If the condition SVO has a participant attached and it is not a pronoun, the full name of participant is entered as the `Who` property. Otherwise, it is left empty.
– SVO labelled with a parallel gateway keyword is treated in a similar way. However, after initialization, the next SVO from a list is added as a parallel task to the current SVO.

[1] For the sake of clarity, we focus on the four main properties of this representation.

- If the SVO is labelled as default flow, the behaviour of function depends on previously detected gateways. If the parallel gateway was found, the SVO is added as another task in this gateway. For the conditional gateway, a default flow is added – an additional task, with keyword `else` as a *Condition* column.
- If none of the previous options were executed, the SVO is added as a simple task, connected by a sequence flow.

After all of the subject-verb-objects constructs are processed, the conditional gateway flag is validated once more – similarly to the last case, if conditional gateway has only one conditional flow, a default flow is added. Finally, the end event is added, which finishes the intermediate process generation.

3.5 BPMN Diagram Generation

The spreadsheet-based model is used to generate a BPMN diagram, using functionality provided by `bpmn_python`[2] – our library written in Python, in order to provide a functionality to import and export BPMN diagrams in the XML format. It provides a functionality for importing spreadsheet-based model description and allows a user to export the imported diagram (stored as a graph structure) into a valid BPMN 2.0 XML file. Using these functionalities, the intermediate model is translated into a diagram, what finishes our process of translating a natural language description into a BPMN diagram.

A prototype of the proposed method was implemented in Python using SpaCy library, which provides Natural Language Processing tools (syntax parser, WordNet lexical database API). This prototype was tested against a test set of natural language business process descriptions, gathered from a few academic sources. One of such examples is presented in the following section.

4 Case Study Example

Let us consider a case study example of computer repair from BPM Academic Initiative[3] presented in Text 1.2.

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

Text 1.2: Text description for the computer repair case study example

[2] See: https://github.com/KrzyHonk/bpmn-python.
[3] See: https://bpmai.org/BPMAcademicInitiative/CreateProcessModels.

Table 5. Spreadsheet-based description for process model obtained from Text 1.2

Order	Activity	Condition	Who
1	Bring defective computer		Customer
2	Check defect		Crs
3a1	Continue process	Costs are acceptable	
3b1	Take her computer home	else	
4	Consist activities		
5	Execute		
6	Check first activity		
7	Configure software		
8	Test proper system functionality		
9a1	Execute arbitrary repair activity	Error is detected	
9b1	Finish repair	else	

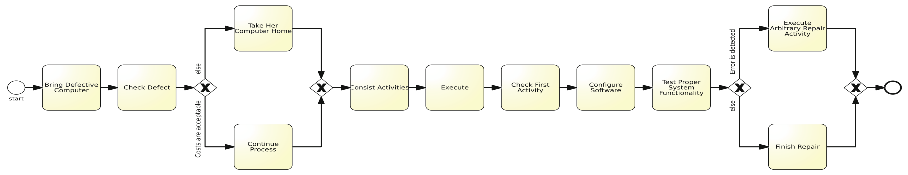


Fig. 6. BPMN process model generated from spreadsheet-based process representation presented in Table 5

For the presented description, our implementation of the algorithm described in Sect. 3 provided the intermediate process model in the spreadsheet-based representation as presented in Table 5 (Fig. 6).

5 Concluding Remarks

In the paper, we present an effective transformation from a semi-formal or informal document into a process model. Such a model can serve as a prototype business process model for further refinement, extension or development.

The proposed solution is based on syntactic analysis of business process description and extracting Subject-Verb-Object constructs, which can be later transformed into process elements. The presented method consists in five steps:

1. Participants extraction.
2. Subject-verb-object constructs extraction.
3. Gateway keywords search.
4. Intermediate process model generation.
5. BPMN diagram generation.

The advantage of our method is the semantic analysis of the text as well as the usage of the intermediate representation, which allows for quick check of the transformation result. Then, the manual correction can be made if necessary. Moreover, as a result we obtain the structured BPMN model, what can help in checking its correctness.

The proposed method of generating process model from natural language description provides some basic information about the described process in the form of BPMN diagram. It is not able to extract more complex constructs and is only able to handle basic elements of BPMN standard. Thus, future works will be focused on enhancing the process models, generated using the proposed method, with additional BPMN elements (such as intermediate events, pools and lanes) as well as adding anaphora resolution to identify real actors in the process. Moreover, we plan to exploit a dedicated domain ontology for exploring related business concepts [15], support verification of the model [16], as well as extend the method to support rules linked to elements of the process [17].

References

1. OMG: Business Process Model and Notation (BPMN) Version 2.0. Technical report, Object Management Group (OMG) (2011)
2. 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 (2017, to appear)
3. Riefer, M., Ternis, S.F., Thaler, T.: Mining process models from natural language text: a state-of-the-art analysis. Multikonferenz Wirtschaftsinformatik (MKWI 2016), pp. 9–11, March 2016
4. Kluza, K., Honkisz, K.: From SBVR to BPMN and DMN models. Proposal of translation from rules to process and decision models. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2016. LNCS (LNAI), vol. 9693, pp. 453–462. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39384-1_39
5. Ghose, A., Koliadis, G., Chueng, A.: Process discovery from model and text artefacts. In: 2007 IEEE Congress on Services, pp. 167–174, July 2007
6. de Almeida Rodrigues Goncalves, J.C., Santoro, F.M., Baiao, F.A.: Business process mining from group stories. In: 13th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2009, pp. 161–166. IEEE (2009)
7. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 482–496. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21640-4_36
8. Kluza, K., Wiśniewski, P., Jobczyk, K., Ligęza, A., Suchenia Mroczek, A.: Comparison of selected modeling notations for process, decision and system modeling. In: FedCSIS 2017, pp. 1095–1098. IEEE (2017)
9. Yue, T., Briand, L.C., Labiche, Y.: An automated approach to transform use cases into activity diagrams. In: Kühne, T., Selic, B., Gervais, M.-P., Terrier, F. (eds.) ECMFA 2010. LNCS, vol. 6138, pp. 337–353. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13595-8_26

10. Njonko, P.B.F., El Abed, W.: From natural language business requirements to executable models via SBVR. In: 2012 International Conference on Systems and Informatics (ICSAI), pp. 2453–2457 (2012)
11. Ferreira, R.C.B., Thom, L.H., de Oliveira, J.P.M., Avila, D.T., dos Santos, R.I., Fantinato, M.: Assisting process modeling by identifying business process elements in natural language texts. In: de Cesare, S., Frank, U. (eds.) ER 2017. LNCS, vol. 10651, pp. 154–163. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70625-2_15
12. Sokolov, K., Timofeev, D., Samochadin, A.: Process extraction from texts using semantic unification. In: KMIS, pp. 254–259 (2015)
13. Kluza, K., Wiśniewski, P.: Spreadsheet-based business process modeling. In: FedCSIS 2016, pp. 1355–1358. IEEE (2016)
14. Muehlen, M., Recker, J.: How much language is enough? Theoretical and practical use of the business process modeling notation. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 465–479. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69534-9_35
15. Nalepa, G., Slazynski, M., Kutt, K., Kucharska, E., Luszpaj, A.: Unifying business concepts for SMEs with Prosecco ontology. In: FedCSIS 2015, pp. 1321–1326 (2015)
16. Klimek, R.: A system for deduction-based formal verification of workflow-oriented software models. Int. J. Appl. Math. Comput. Sci. **24**(4), 941–956 (2014)
17. Wang, W., Indulska, M., Sadiq, S.: Guidelines for business rule modeling decisions. J. Comput. Inf. Syst. **58**(4), 363–373 (2018)