Process Model Generation from Natural Language Text

Fabian Friedrich[1], Jan Mendling[2], and Frank Puhlmann[1]

[1] inubit AG, Schöneberger Ufer 89-91, 10785 Berlin, Germany
{Fabian.Friedrich,Frank.Puhlmann}@inubit.com
[2] Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany
jan.mendling@wiwi.hu-berlin.de

Abstract. Business process modeling has become an important tool for managing organizational change and for capturing requirements of software. A central problem in this area is the fact that the acquisition of as-is models consumes up to 60% of the time spent on process management projects. This is paradox as there are often extensive documentations available in companies, but not in a ready-to-use format. In this paper, we tackle this problem based on an automatic approach to generate BPMN models from natural language text. We combine existing tools from natural language processing in an innovative way and augmented them with a suitable anaphora resolution mechanism. The evaluation of our technique shows that for a set of 47 text-model pairs from industry and textbooks, we are able to generate on average 77% of the models correctly.

1 Introduction

Business process management is a discipline which seeks to increase the efficiency and effectiveness of companies by holistically analyzing and improving business processes across departmental boundaries. In order to be able to analyze a process, a thorough understanding of it is required first. The necessary level of insight can be obtained by creating a formal model for a given business process.

The required knowledge for constructing process models has to be made explicit by actors participating in the process [1]. However, these actors are usually not qualified to create formal models themselves [2]. For this reason, modeling experts are employed to iteratively formalize and validate process models in collaboration with the domain experts. This traditional procedure of extracting process models involves interviews, meetings, or workshops [3]. It entails considerable time and costs due to ambiguities or misunderstandings between the involved participants [4]. Therefore, the initial elicitation of conceptual models is considered to be a knowledge acquisition bottleneck [5]. According to Herbst [1] the acquisition of the as-is model in a workflow project requires 60% of the total time spent. Accordingly, substantial savings are possible by providing appropriate tool support to speed up the acquisition phase.

H. Mouratidis and C. Rolland (Eds.): CAiSE 2011, LNCS 6741, pp. 482–496, 2011.

In this context, it is a paradox that acquisition is costly although detailed information about processes is often already available in the form of informal textual specifications. Such textual documents can be policies, reports, forms, manuals, content of knowledge management systems, and e-mail messages. Content management professionals estimated that 85% of the information in companies is stored in such an unstructured format [6]. Moreover, the amount of unstructured text is growing at a much faster rate than structured data [7]. It seems reasonable to assume that these texts are relevant sources of information for the construction of conceptual models.

In this paper, we develop an approach to directly extract business process models from textual descriptions. Our contribution is a corresponding technique that does not make any assumptions about the structure of the provided text. We combine an extensive set of tools from natural language processing (NLP) in an innovative way and augment it with an anaphora resolution mechanism, which was particularly developed for our approach. The evaluation of our technique with a set of 47 text-model pairs from industry and textbooks reveals that on average 77% of the model is correctly generated. We furthermore discuss current limitations and directions of improvement.

The paper is structured as follows. Section 2 introduces the foundations of our approach, namely BPMN process models and natural language processing techniques. Section 3 identifies a set of language processing requirements, and illustrates how they are tackled in the various steps of our generation approach. Section 4 presents our evaluation results based on a sample of text-model pairs. Section 5 discusses related work before Section 6 concludes the paper.

2 Background

Generating models builds on understanding the essential concepts of BPMN process models and of state-of-the-art techniques for natural language processing. In this section, we introduce BPMN and then natural language processing tools.

The Business Process Model and Notation (BPMN) is a standard for process modeling that has been recently published in its version 2.0 [8]. It includes four categories of elements, namely Flow Objects (Activities, Events and Gateways), Swimlanes (Pools and Lanes), Artifacts (e.g. Data Objects, Text Annotations or Groups), and Connecting Objects (Sequence Flows, Message Flows and Associations). The first three are nodes, the latter ones are edges. Figure 1 shows a BPMN example of a claims handling process provided by QUT. The process is subdivided into three pools (one with two lanes) capturing the actors of the process. Activities are depicted as rounded boxes. Different events (round elements with icons) for sending and receiving messages affect the execution of the process. The diamond-shaped elements define specific routing behavior as gateways.

A BPMN process model is typically the result of analyzing textual descriptions of a process. A claims handling process provided by QUT is described as follows: “The process starts when a customer submits a claim by sending in relevant documentation. The Notification department at the car insurer checks

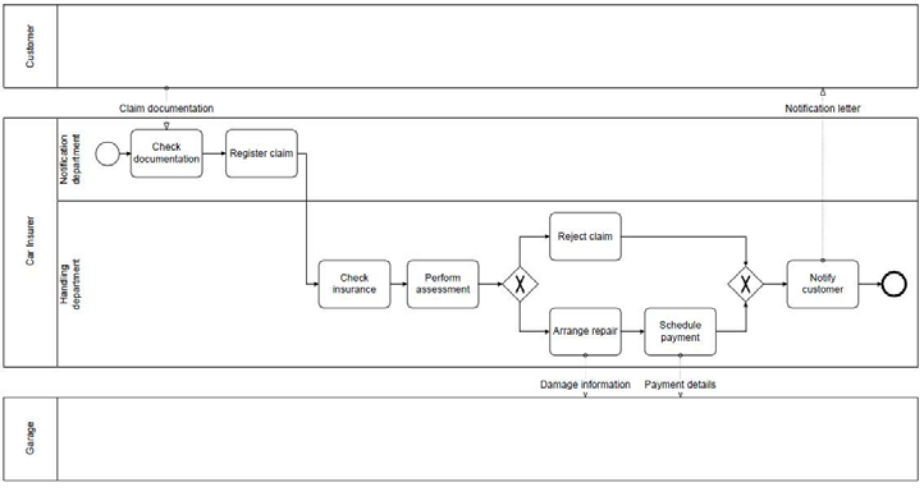

Fig. 1. Example of a claims handling process in BPMN

the documents upon completeness and registers the claim. Then, the Handling department picks up the claim and checks the insurance. Then, an assessment is performed. If the assessment is positive, a garage is phoned to authorise the repairs and the payment is scheduled (in this order). Otherwise, the claim is rejected. In any case (whether the outcome is positive or negative), a letter is sent to the customer and the process is considered to be complete." Such information is usually provided by people working in the process and then formalized as a model by system analysts [2].

For our model generation approach, we will employ methods from computational linguistics and natural language processing. This branch of artificial intelligence deals with analyzing and extracting useful information from natural language texts or speech. For our approach, three concepts are of vital importance: syntax parsing, which is the determination of a syntax tree and the grammatical relations between the parts of the sentence; semantic analysis, which is the extraction of the meaning of words or phrases; and anaphora resolution, which involves the identification of the concepts which are references using pronouns ("we","he","it") and certain articles ("this", "that"). For syntax parsing and semantic analysis, there are standard tools available.

The Stanford Parser is a syntax parsing tool for determining a syntax tree. This tree shows the dependencies between the words of the sentence through the tree structure [10]. Additionally, each word and phrase is labeled with an appropriate part-of-speech and phrase tag. The tags of the *Stanford Parser* are the same which can be found in the *Penn Tree Bank* [11]. The *Stanford Parser* also produces 55 different *Stanford Dependencies* [12]. These dependencies reflect the grammatical relationships between the words. Such grammatical relations provide an abstraction layer to the pure syntax tree. They also contain information about the syntactic role of all elements.

There are also tools available for semantic analysis. They provide semantic relations on different levels of detail. We use FrameNet [13] and the lexical database WordNet[14]. WordNet provides various links to synonyms, homonyms, and hypernyms for a particular class of meaning associated with a synonym-set. FrameNet defines semantic relations that are expected for specific words. These relations are useful, e.g., to recognize that a verb "send" would usually go with a particular object being sent. Syntax parsers and semantic analysis are used in our transformation approach, augmented with anaphora resolution.

3 Transformation Approach

The most important issue we are facing when trying to build a system for generating models is the complexity of natural language. We collected issues related to the structure of natural language texts from the scientific literature and analyzed the test data, which is described in section 4. Thereby, we were able to identify four broad categories of issues which we have to solve in order to analyze natural language process descriptions successfully (see Table 1). *Syntactic Leeway* relates to the fact that there is a mismatch between the semantic and syntactic layer of a text. *Atomicity* deals with the question of how to construct a proper phrase-activity mapping. *Relevance* has to check whether parts of the text might be irrelevant for the generated process model. Finally, *Referencing* addresses the question of how to resolve relative references between words and between sentences.

Table 1. References in the literature to the analyzed issues

Issue	Refs.	Issue	Refs.
1 Syntactic Leeway		3 Relevance	
1.1 Active-Passive	[15]	3.1 Relative Clause Importance	[16]
1.2 Rewording/Order	[17,18]	3.2 Example Sentences	[19]
1.3 Implicit Conditions	[20,21]	3.3 Meta-Sentences	[16]
2 Atomicity		4 Referencing	
2.1 Complex Sentences	[16,18]	4.1 Anaphora	[22,23]
2.2 Action Split over Sentences	[22]	4.2 Textual Links	[24]
2.3 Relative Clauses	[16]	4.3 End-of-block Recognition	[19,16]

Different solution strategies were applied in the works listed in Table 1 to overcome the stated problems, e.g. by constricting the format of the textual input [17], but no study considers all mentioned problems and offers a comprehensive solution strategy. Another interesting fact is that none of the works using a shallow parser shows how they deal with passive voice [15,22,23,17]. We solved this problem by using the grammatical relations of the Stanford Parser.

To obtain a structured representation of the knowledge we extract from the text, we decided to store it in a *World Model*, as opposed to a direct straight

through model generation. This approach was also taken by most of the other works which built a similar system [17,22,18,23]. The data structure used by the approach of the University of Rio de Janeiro [23] was taken from the CREWS project [15]. The authors argue that it is suited well for this task as a scenario description corresponds to the description of a process model. Therefore, we also use the CREWS scenario metamodel as starting point. However, we modified several parts as, e.g., we explicitly represent connections between the elements using the class "Flow". Additionally, we explicitly considered traceability as a requirement. Thus, attributes relating an object to a sentence or a word are added to the World Model. The four main elements of our World Model are Actor, Resource, Action, and Flow. This World Model will be used throughout all phases of our transformation procedure to capture syntactic and semantic analysis results. Each phase is allowed to access, modify and add data.

The rest of this section is dedicated to analyzing and discussing the issues collected in Table 1. We will then seize the developed suggestions and reference these issues during the description of our transformation approach. Section 3.1 discusses sentence level analysis for finding actions. Section 3.2 investigates text level analysis for enriching the data stored in the world model. Finally, Section 3.3 describes the generation of a BPMN model. While we focus on the general procedure here, we documented details of all algorithms in [25].

3.1 Sentence Level Analysis

The first step of our transformation procedure is a sentence level analysis. The extraction procedure consists of the steps that are outlined as a BPMN model in Figure 2. This overview also shows the different components upon which our transformation procedure builds and their usage of Data Sources.

The text is processed in several stages. First, a tokenization splits up the text into individual sentences. The challenge here is to distinguish a period used for an abbreviation (e.g. M.Sc.) from a period marking the end of a sentence.

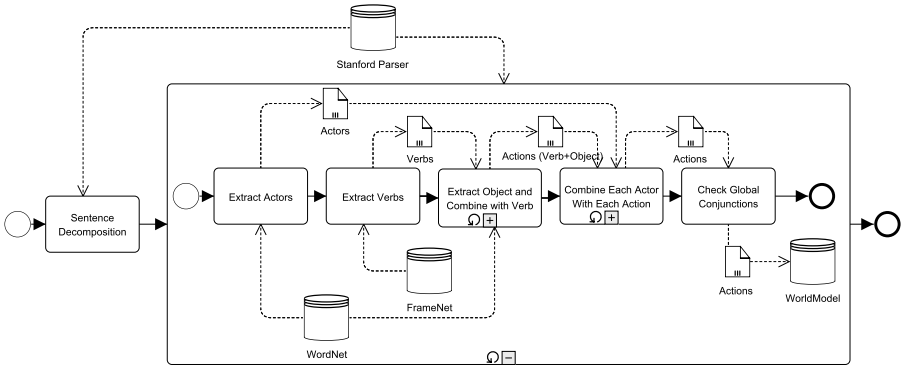

Fig. 2. Structural overview of the steps of the Sentence Level Analysis

Afterwards, each sentence is parsed by the *Stanford Parser* using the factored model for English [11]. We utilize the factored model and not the pure probabilistic context free grammar, because it provides better results in determining the dependencies between markers as "if" or "then", which are important for the process model generation. Next, complex sentences are split into individual phrases. This is accomplished by scanning for sentence tags on the top level of the Parse Tree and within nested prepositional, adverbial, and noun phrases.

Once the sentence is broken down into individual constituent phrases, actions can be extracted. First, we determine whether the *parsedSentence* is in active or passive voice by searching for the appropriate grammatical relations (Issue 1.1). Then, all Actors and Actions are extracted by analyzing the grammatical relations. To overcome the problem of example sentences mentioned earlier (Issue 3.2) the actions are also filtered. This filtering method simply checks whether the sentence contains a word of a stop word list called *example indicators*. Then, we extract all objects from the phrase and each Action is combined with each Object. The same is done with all Actors. This procedure is necessary as an Action is supposed to be atomic according to the BPMN specification [8] and Issue 2.1. Therefore, a new Action has to be created for each piece of information as illustrated in the following example sentences. In each sentence the conjunction relation which causes the extraction of several Actors, Actions or Resources is highlighted. As a last step, all extracted Actions are added to the World Model.

- "Likewise the old supplier **creates and sends** the final billing to the customer." (Action)
- "It is given either by **a sales representative or by a pre-sales employee** in case of a more technical presentation." (Actor)
- "At this point, the Assistant Registry Manager puts **the receipt and copied documents** into an envelope and posts it to the party." (Resource)

3.2 Text Level Analysis

This section describes the text level analysis. It analyzes the sentences taking their relationships into account. The structural overview of this phase is shown in Figure 3. We use the Stanford Parser and WordNet here, and also an anaphora resolution algorithm. During each of the five steps, the Actions previously added to the World Model are augmented with additional information.

An important part of the algorithm presented here is the determination heuristic for resolving relative references within the text (Issue 4.1). Existing libraries are not seamlessly integrateable with the output provided by the Stanford Parser. Therefore, we implemented a simple anaphora resolution technique for the resolution of determiner and pronouns. This procedure is described in detail in [25]. An experimental evaluation using our test data set showed that this approach achieved a good accuracy of 63.06%.

The second step in our analysis is the detection of conditional markers. These markers can either be a single word like "if", "then", "meanwhile" or "otherwise", or a short phrase like "in the meantime" or "in parallel". All of these

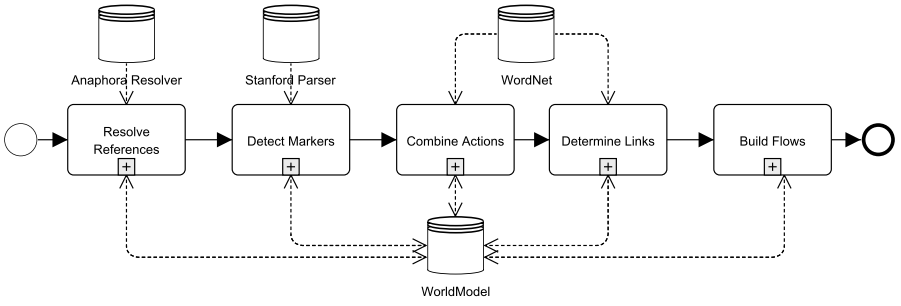

Fig. 3. Structural overview of the steps of the Text Level Analysis

markers have specific characteristics and can be mapped to different BPMN constructions. In order to capture this semantic information we compiled four lists, namely ConditionIndicators (exclusive gateway), ParallelIndicators (parallel gateway), ExceptionIndicators (for Error Intermediate Events), and SequenceIndicators (for the continuation of a branch of a gateway). These lists do not claim completeness and can be extended by the user, if necessary.

We can use the information gathered so far to combine the information contained in two different Actions. This procedure tackles the problem of Actions which are split up over several sentences (Issue 2.2). To consider two Actions as a candidate for a merger, a reference had to be established between them during the anaphora resolution phase. This reference can either directly point from the Actor or from the Object of this Action. But, for the case that the Object points to another Actor or Resource we also consider the Action which contains it as a possible candidate. Next, it is checked whether the objects can be merged by checking various characteristics of both Actions. If the actions truly complement each other, they can be merged and form one single action. When both Actions complement each other except for the negation modifier we can still enhance the information content of one action by copying information, as the initiating Actor, the Object, and/or the copula attribute. An example for such a case are these sentences: "Of course, asking the customer whether he is generally interested is also important." and "If this is not the case, we leave him alone, [...]"

For Issue 4.2, we defined three types of textual references: forward, backward, and jump references. In order to identify those links in the text automatically, we start by comparing all actions within our World Model to one another. It is then determined whether the selected actions can be linked or not. Within this method, we compare the following characteristics of both Actions: Copula Specifier, Negation Status, the initiating Actor (ActorFrom), the Object, the open clausal complement, and the Prepositional Specifiers, whose head word is "to" or "about". The elements are compared using their root form provided by WordNet. If the elements differ or an element is defined for one Action, but not for the other, the Actions cannot be merged. Otherwise, the Actions are considered equal and a link relationship can be established. Additionally, the type of the link relationship is determined and saved along with the link.

The last step of the text level analysis is the generation of Flows. A flow describes how activities are interacting with each other. Therefore, during the process model generation such Flows can be translated to BPMN connecting objects. When creating the Flows we build upon the assumption that a process is described sequentially and upon the information gathered in the previous steps. The word, which connected the items is important to determine how to proceed and what type of gateway we have to create. So far we support a distinction between "or", "and/or", and "and". Other conjunctions are skipped.

3.3 Process Model Generation

In the last phase of our approach the information contained in the World Model is transformed into its BPMN representation. We follow a nine step procedure, as depicted in Figure 3.3. The first 4 steps: creation of nodes, building of Sequence Flows, removal of dummy elements, the finishing of open ends, and the processing of meta activities are used to create an initial and complete model. Optionally, the model can be augmented by creating Black Box Pools and Data Objects. Finally, the model is laid out to achieve a human-readable representation.

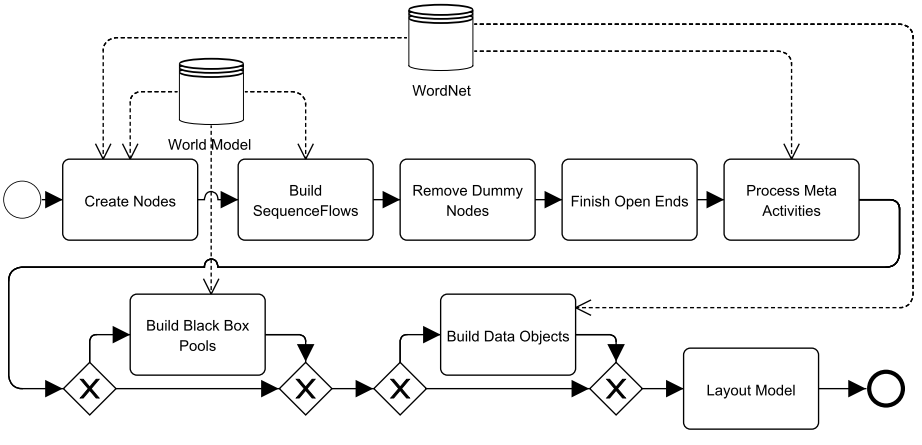

Fig. 4. Structural overview of the steps of the Process Model Generation phase

The first step required for the model creation is the construction of all nodes of the model. After the Flow Object was generated, we create a Lane Element representing the Actor initiating the Action. If no Lane was determined for an Action, it is added to the last Lane which was created successfully as we assume that the process is described in a sequential manner. The second step required during the model creation is the construction of all edges. Due to the definition of Flows within our World Model, this transformation is straight-forward. Whenever a Flow which is not of the type "Sequence" is encountered, a Gateway appropriate for the type of the flow is created. An exception to that is the

type "Exception". If the World Model contains a flow of this type, an exception intermediate event is attached to the task which serves as a source and this Intermediate Event is connected to the target instead of the node itself. We then skip dummy actions, which were inserted between gateways directly following each other.

Step four is concerned with open ends. So far, no Start and End Events were created. This is accomplished in this step. The procedure is also straight forward. We create a preceding Start event to all Tasks which do not have any predecessors (in-flow = 0) and succeeding End Events to all Tasks which do not have any successors (out-flow = 0). Additionally, Gateways whose in- and out-flow is one receive an additional branch ending in an End Event.

The last step in the model creation phase handles Meta-Activities (Issue 3.3). We search and remove redundant nodes directly adjacent to Start or End Events. This is required as several texts contain sentences like "[...] the process flow at the customer also ends." or "The process of "winning" a new customer ends here." If such sentences are not filtered, we might find tasks labeled "process ends" right in front of an end event or "start workflow" following a start event. We remove nodes whose verb is contained in the hypernym tree of "end" or "start" in WordNet if they are adjacent to a Start or End Event.

The execution of these five steps yields a full BPMN model. As the elements of this model do not contain any position information yet, our generation procedure concludes with an automated layout algorithm. We utilize a simple grid layout approach similar to [26], enhanced with standard layout graph layout algorithms as Sugiyama [27] and the topology-shape-metric approach [28]. For the example text of the claims handling process from Section 2 we generated the model given in Figure 5. The question of how far this result can be considered to be accurate is discussed in the following section.

4 Evaluation of Generated Process Models

For the validation of our approach, we collected a test data set consisting of 47 of those text-model pairs, each including a textual process description and a corresponding BPMN models created by a human modeler. Different sources from research and practice were incorporated into our test data set: Academic (15 models), Industry (9 models), Textbook (9 models), and Public Sector (14 models), see Table 2. While the academic pairs were provided by our university partners, the industry models are taken from two main sources. First, we gathered four models from the websites of three BPM tool vendors, namely Active VOS, Oracle, and BizAgi. Four models stem from training material of inubit AG, another one from a German BPM practitioner, and further ones from two BPMN textbooks [29,9]. Finally, we included the definition of switch processes of the *Federal Network Agency* of Germany in its semi-structured tabular format and the corresponding model.

To avoid unintended effects while parsing, minor corrections were applied to the texts. Some models were translated, some were converted from other

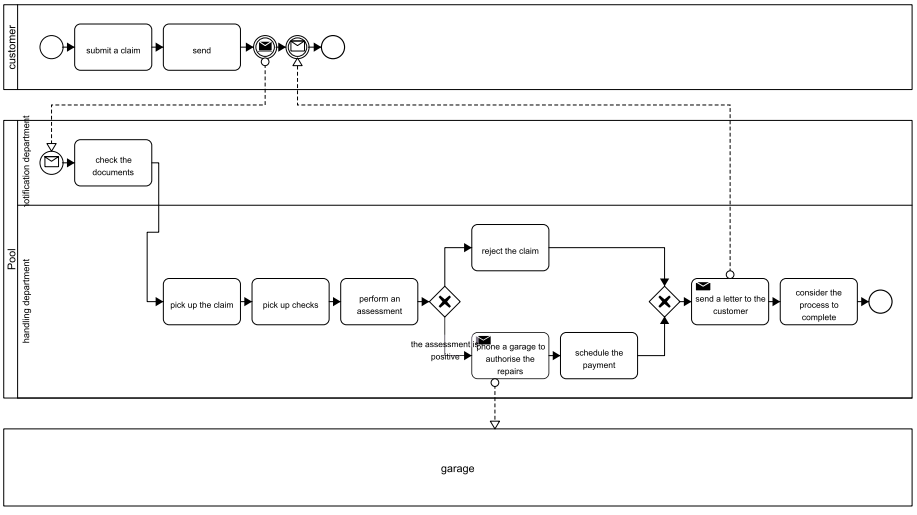

Fig. 5. The claims handling model as generated by our system

modeling languages to BPMN in order to compare them. Table 2 lists characteristics of the texts and models of our data set. The table captures the following data: a unique ID, the number of models (M), the number of sentences (n), the average length of sentences (∅l), the size of the models in terms of nodes ($|N|$), gateways ($|G|$), and edges ($|E|$). All our material is published in [25].

The evaluation results are based on the similarity (sim) between the manually and automatically created models. We employ the metric of *Graph Edit Distance*. To compute the Graph Edit Distance, the graph representation of the process models is analyzed. The labels, attributes, the structural context, and behavior are compared [32]. Afterwards a greedy graph matching heuristic [33] is employed

Table 2. Characteristics of the test data set by source (average values)

ID	Source	M	Type	n	∅l	$\|N\|$	$\|G\|$	$\|E\|$
1	HU Berlin	4	academic	10.00	18.14	25.75	6.00	24.50
2	TU Berlin [30]	2	academic	34.00	21.17	71.00	9.50	79.50
3	QUT	8	academic	6.13	18.26	14.88	1.88	16.00
4	TU Eindhoven [31]	1	academic	40.00	18.45	38.00	8.00	37.00
5	Vendor Tutorials	4	industry	9.00	18.20	14.00	2.25	11.50
6	inubit AG	4	industry	11.50	18.38	24.00	4.25	21.25
7	BPM Practicioners	1	industry	7.00	9.71	13.00	1.00	9.00
8	BPMN Prac. Handbook [9]	3	textbook	4.67	17.03	13.00	1.33	14.67
9	BPMN M&R Guide [29]	6	textbook	7.00	20.77	23.83	3.00	23.67
10	FNA - Metrology Processes	14	public sector	6.43	13.95	24.43	3.14	25.93
	Total	**47**		**9.19**	**17.16**	**23.21**	**3.38**	**23.64**

to create pairs of nodes and edges. We use the greedy heuristic as it showed the best performance without considerable accuracy trade-offs. After the mapping is created, a Graph Edit Distance value can be calculated given:

- N_i - set of nodes in model i
- E_i - set of edges of model i
- $\overline{N_i}$ - the set of nodes in model i which were not mapped
- $\overline{E_i}$ - the set of edges in model i which were not mapped
- M - The mapping between the nodes of model 1 and 2

An indicator for the difference between the models can be calculated as:

$$m^* = \begin{cases} \sum_{i=1}^{|M|} 1 - sim(M_i) & \text{if} |M| > 0 \\ 1.0 & \text{otherwise} \end{cases} \quad (1)$$

As a last step weights for the importance of the differences (w_{map}), the unmapped Nodes (w_{uN}), and the unmapped Edges (w_{uE}) have to be defined. For our experiments we gave the difference a slightly higher importance and assigned $w_{\text{map}} = 0.4$ and $w_{\text{uN}} = w_{\text{uE}} = 0.3$. The overall graph edit distance then becomes:

$$\text{sim}(m_1, m_2) = 1 - (w_{\text{map}} * \frac{m^*}{|M|} + w_{\text{uN}} * \frac{\overline{|N_1|} + \overline{|N_2|}}{|N_1| + |N_2|} + w_{\text{uE}} * \frac{\overline{|E_1|} + \overline{|E_2|}}{|E_1| + |E_2|}) \quad (2)$$

This value ranges between 0 and 1. For the case that all nodes could be mapped with a similarity of 1.0 the terms will also become 1.0. If the mapping is not optimal, the term in parenthesis will grow steadily and the similarity decreases. If no nodes are mapped at all, the similarity will be 0.

For our evaluation, we generated the model for each text and calculated the similarity metric between it and the original BPMN model. The results are shown in Table 3. Columns 2-4 show that the concepts of meta sentences, relative references, and textual jumps are important for almost all elements within our test data. The following six columns show the average values of nodes, gateways,

Table 3. Result of the application of the evaluation metrics to the test data set

ID	m	r	j	$\lvert N_{\text{gen}}\rvert$	$\Delta\lvert N_{\text{gen}}\rvert$	$\lvert G_{\text{gen}}\rvert$	$\Delta\lvert G_{\text{gen}}\rvert$	$\lvert E_{\text{gen}}\rvert$	$\Delta\lvert E_{\text{gen}}\rvert$	sim
1	3	5,25	0	30,25	14,88%	5,50	-9,09%	28,75	14,78%	77,94%
2	7,50	7,50	2,50	91,50	22,40%	13,00	26,92%	94,00	15,43%	70,79%
3	0,50	1,38	0,00	20,25	26,54%	2,63	28,57%	20,13	20,50%	78,78%
4	8,00	4,00	1,00	63,00	39,68%	1,00	-700,00%	52,00	28,85%	41,54%
5	1,25	1,75	1,75	24,75	43,43%	4,25	47,06%	23,00	50,00%	63,63%
6	2,25	8,00	0,50	29,75	19,33%	2,75	-54,55%	25,25	15,84%	60,93%
7	0,00	5,00	0,00	14,00	7,14%	2,00	50,00%	11,00	18,18%	74,35%
8	0,00	5,00	0,33	13,33	2,50%	1,00	-33,33%	10,33	-41,94%	77,49%
9	0,83	1,50	0,33	22,33	-6,72%	3,33	10,00%	20,83	-13,60%	71,77%
10	0,00	0,21	0,36	25,29	3,39%	3,71	15,38%	27,29	4,97%	89,81%
Total	**1,23**	**2,60**	**0,49**	**27,43**	**15,36%**	**3,72**	**9,14%**	**26,77**	**11,69%**	**76,98%**

and edges within the generated models. We can see that the transformation procedure tends to produce models which are on average 9-15% larger in size then what a human would create. This can be partially explained by noise and meta sentences which were not filtered appropriately. On the other hand, humans tend to abstract during the process of modeling. Therefore, we often find more detail of the text also in the generated model. The results are highly encouraging as our approach is able to correctly recreate 77% of the model in average. On a model level up to 96% of similarity can be reached, which means that only minor corrections by a human modeler are required.

During the detailed analysis we determined different sources of failure, which resulted in a decreased metric value. These are noise, different levels of abstractions, and processing problems within our system. *Noise* includes sentences or phrases that are not part of the process description, as for instance "This object consists of data elements such as the customers name and address and the assigned power gauge." While such information can be important for the understanding of a process, it leads to unwanted Activities within the generated model. To tackle this problem, further filtering mechanisms are required. Low similarity also results from *difference in the level of granularity*. To solve this problem, we could apply automated abstraction techniques like [34] on the generated model. Finally, the employed *natural language processing components failed* during the analysis. At stages, the Stanford Parser failed at correctly classifying verbs. For instance, the parser classified "the second activity checks and configures" as a noun phrase, such that the verbs "check" and "configure" cannot be extracted into Actions. Furthermore, important verbs related to business processes are not contained in FrameNet, as "report". Therefore, no message flow is created between report activities and a Black Box Pool. We expect this problem to be solved in the future as the FrameNet database grows. With WordNet, for instance, there is a problem with times like "2:00 pm", where pm as an abbreviation for "Prime Minister" is classified as an Actor. To solve this problem a reliable sense disambiguation has to be conducted. Nevertheless, overall good results were achieved by using WordNet as a general purpose Ontology.

5 Related Work

Recently, there is an increasing interest in the derivation of conceptual models from text. This research is mainly conducted by six different groups.

Two approaches generate UML models. The Klagenfurt Conceptual Pre-design Model and a corresponding tool are used to parse German text and fill instances of a generic meta-model [35]. The stored information can be transformed to UML activity diagrams and class diagrams [18]. The transformation from text to the meta-model requires the user to make decisions about the relevant parts of a sentence. In contrast to that, the approach described in [36] is fully automated. It uses use-case descriptions in a format called RUCM to generate activity diagrams and class diagrams [17]. Yet, the system is not able to parse free-text. The RUCM input is required to be in a restricted format allowing only 26 types

of sentence structures, which rely on keywords like "VALIDATES THAT" or "MEANWHILE". Therefore, it can hardly be used in the initial process definition phase as it would require rewriting of process-relevant documents.

The University of Rio de Janeiro focuses on the derivation of BPMN models from group stories provided in Portuguese [23]. The approach was tested with a course enrollment process modeled by students. The examples in their paper show that process models can be created successfully, but a couple of their exhibits show that syntactical problems can occur, e.g. implicit conditions, which we explicitly tackle with our approach. The *R-BPD* toolkit from the University of Wollongong uses a syntax parser to identify verb-object phrases [21]. It also identifies textual patterns like "If <condition/event>, [then] <action>" [20]. The result are rather BPMN snippets than fully connected models. Nevertheless, this toolkit is able to take existing models into account for cross validation.

A fifth approach is the one of Policy-Driven Process Mapping [37]. First, a procedure was developed which creates a BPMN diagram, given that data items, tasks, resources (actors), and constraints are identified in an input text document. Although the approach does not require a process description to be sequential, it does not support Pools, Data Objects, and Gateways other than an exclusive split. Furthermore, user-interaction is required at several stages.

The approach by Sinha et al. builds on a linguistic analysis pipeline [22,38]. First, text is preprocessed with a part-of-speech tagger. Next, words are annotated with dictionary concepts, which classify verbs using a domain ontology. Then, an anaphora resolution algorithm and a context annotator are applied. The resulting information is then transferred to a Use Case Description metamodel and later into a BPMN process model. The dictionary concepts, which are a vital part of their approach, rely on a domain ontology which has to be hand-crafted. This imposes a manual effort when transferring the system to other types of texts or languages. Instead, our approach builds on the free WordNet and FrameNet lexical databases, which are available for different languages.

6 Conclusion

In this paper, we presented an automatic approach to generate BPMN models from natural language text. We have combined existing tools from natural language processing in an innovative way and augmented them with a suitable anaphora resolution mechanism. The evaluation of our technique shows that for a set of 47 text-model pairs from industry and textbooks, we are able to generate on average 77% of the models correctly.

Despite these encouraging results, we still require empirical user studies. Such studies should investigate whether humans find the generated models useful and easy to adapt towards a fully accurate model. Furthermore, our system is able to read process descriptions consisting of full sentences. Furthermore, we assumed the description to be sequential and to contain no questions and little process-irrelevant information. Another prerequisite is that the text is grammatically correct and constituent. Thus, the parsing of structured input, like tables or

texts making use of indentions, or texts which are of low quality is not possible at the moment and presents opportunities for further research.

While the evaluation conducted in this thesis evinced encouraging results different lines of research could be pursued in order to enhance the quality or scope of our process model generation procedure. As shown the occurrence of meta-sentences or noise in general is one of the severest problems affecting the generation results. Therefore, we could improve the quality of our results by adding further rules and heuristics to identify such noise. Another major source of problems was the syntax parser we employed. As an alternative, semantic parsers like [39] could be investigated.

References

1. Herbst, J., Karagiannis, D.: An inductive approach to the acquisition and adaptation of workflow models. In: Proceedings of the IJCAI, pp. 52–57 (1999)
2. Frederiks, P., Van der Weide, T.: Information modeling: the process and the required competencies of its participants. Data & Knowledge Engineering 58(1), 4–20 (2006)
3. Scheer, A.: ARIS-business process modeling. Springer, Heidelberg (2000)
4. Reijers, H., Limam, S., Van Der Aalst, W.: Product-based workflow design. Journal of Management Information Systems 20(1), 229–262 (2003)
5. Gruber, T.: Automated knowledge acquisition for strategic knowledge. Machine Learning 4(3), 293–336 (1989)
6. Blumberg, R., Atre, S.: The problem with unstructured data. DM Review 13, 42–49 (2003)
7. White, M.: Information overlook. EContent(26:7) (2003)
8. OMG, eds.: Business Process Model and Notation (BPMN) Version 2.0 (June 2010)
9. Freund, J., Rücker, B., Henninger, T.: Praxishandbuch BPMN. Hanser (2010)
10. Melčuk, I.: Dependency syntax: theory and practice, New York (1988)
11. Marcus, M., Marcinkiewicz, M., Santorini, B.: Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics 19(2), 330 (1993)
12. de Marneffe, M., Manning, C.: The Stanford typed dependencies representation. In: Workshop on Cross-Framework and Cross-Domain Parser Evaluation, pp. 1–8 (2008)
13. Baker, C., Fillmore, C., Lowe, J.: The berkeley framenet project. In: 17th Int. Conf. on Computational Linguistics, pp. 86–90 (1998)
14. Miller, G.A.: Wordnet: A lexical database for english. CACM 38(11), 39–41 (1995)
15. Achour, C.B.: Guiding scenario authoring. In: 8th European-Japanese Conference on Information Modelling and Knowledge Bases, pp. 152–171. IOS Press, Amsterdam (1998)
16. Li, J., Wang, H., Zhang, Z., Zhao, J.: A policy-based process mining framework: mining business policy texts for discovering process models. ISEB 8(2), 169–188
17. Yue, T., Briand, L., Labiche, Y.: An Automated Approach to Transform Use Cases into Activity Diagrams. Modelling Foundations and Appl., 337–353 (2010)
18. Fliedl, G., Kop, C., Mayr, H., Salbrechter, A., Vöhringer, J., Weber, G., Winkler, C.: Deriving static and dynamic concepts from software requirements using sophisticated tagging. Data & Knowledge Engineering 61(3), 433–448 (2007)
19. Kop, C., Mayr, H.: Conceptual predesign–bridging the gap between requirements and conceptual design. In: 3rd Int. Conf. on Requirements Eng. p. 90 (1998)

20. Ghose, A., Koliadis, G., Chueng, A.: Process Discovery from Model and Text Artefacts. In: 2007 IEEE Congress on Services, pp. 167–174 (2007)
21. Ghose, A.K., Koliadis, G., Chueng, A.: Rapid business process discovery (R-BPD). In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 391–406. Springer, Heidelberg (2007)
22. Sinha, A., Paradkar, A., Kumanan, P., Boguraev, B.: An Analysis Engine for Dependable Elicitation on Natural Language Use Case Description and its Application to Industrial Use Cases. Technical report, IBM (2008)
23. de AR Gonçalves, J.C., Santoro, F.M., Baião, F.A.: A case study on designing processes based on collaborative and mining approaches. In: Int. Conf. on Computer Supported Cooperative Work in Design, Shanghai, China (2010)
24. Fliedl, G., Kop, C., Mayr, H.: From textual scenarios to a conceptual schema. Data & Knowledge Engineering 55(1), 20–37 (2005)
25. Friedrich, F.: Automated generation of business process models from natural language input. Master's thesis, Humboldt-Universität zu Berlin (November 2010)
26. Kitzmann, I., Konig, C., Lubke, D., Singer, L.: A Simple Algorithm for Automatic Layout of BPMN Processes. In: IEEE Conf. CEC, pp. 391–398 (2009)
27. Seemann, J.: Extending the sugiyama algorithm for drawing UML class diagrams: Towards automatic layout of object-oriented software diagrams. In: Graph Drawing, pp. 415–424. Springer, Heidelberg (1997)
28. Eiglsperger, M., Kaufmann, M., Siebenhaller, M.: A topology-shape-metrics approach for the automatic layout of UML class diagrams. In: Proceedings of the 2003 ACM Symposium on Software Visualization, p. 189. ACM, New York (2003)
29. White, S., Miers, D.: BPMN Modeling and Reference Guide: Understanding and Using BPMN. Future Strategies Inc. (2008)
30. Holschke, O.: Impact of granularity on adjustment behavior in adaptive reuse of business process models. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 112–127. Springer, Heidelberg (2010)
31. Reijers, H.: Design and control of workflow processes: business process management for the service industry. Eindhoven University Press (2003)
32. Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Inf. Sys. 36, 498–516 (2010)
33. Dijkman, R., Dumas, M., Garcıa-Banuelos, L., Käärik, R.: Graph Matching Algorithms for Business Process Model Similarity Search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 48–63. Springer, Heidelberg (2009)
34. Polyvyanyy, A., Smirnov, S., Weske, M.: On application of structural decomposition for process model abstraction. In: 2nd Int. Conf. BPSC, pp. 110–122 (March 2009)
35. Kop, C., Vöhringer, J., Hölbling, M., Horn, T., Irrasch, C., Mayr, H.: Tool Supported Extraction of Behavior Models. In: Proc. 4th Int. Conf. ISTA (2005)
36. Yue, T., Briand, L., Labiche, Y.: Automatically Deriving a UML Analysis Model from a Use Case Model. Technical report, Carleton University (2009)
37. Wang, H.J., Zhao, J.L., Zhang, L.J.: Policy-Driven Process Mapping (PDPM): Discovering process models from business policies. DSS 48(1), 267–281 (2009)
38. Sinha, A., Paradkar, A.: Use Cases to Process Specifications in Business Process Modeling Notation. In: 2010 IEEE Int. Conf. on Web Services, pp. 473–480 (2010)
39. Shi, L., Mihalcea, R.: Putting Pieces Together: Combining FrameNet, VerbNet and WordNet for Robust Semantic Parsing. In: Proceedings of the 6th Int. Conf. on Computational Linguistics and Intelligent Text Processing, p. 100 (2005)