An Approach to Support Process Model Validation based on Text Generation

Martin Meitz[1], Henrik Leopold[2], Jan Mendling[1]
[1]Wirtschaftsuniversität Wien, Austria
[jan.mendling]@wu.ac.at
[2] Humboldt-Universität zu Berlin, Germany
henrik.leopold@wiwi.hu-berlin.de

Abstract: In many organizations business process modeling plays an important role for understanding, documenting and also redesigning operations. However, often the validation and the usage of process models is hampered by the fact that many domain experts or not able to understand the models in detail. In order to overcome this problem, we propose an approach for automatically generating description texts from Petri Nets. Our technique combines different techniques from linguistics and graph decomposition in order to generate accurate natural language texts. Further, we support our technique with insights from the multimedia learning theory which explicitly recommends the addressing of different channels such as the auditory and the visual channel. As a proof of concept we implemented our technique in the open source modeling tool WoPeD.

1 Introduction

Business process modeling plays an important role both in business process redesign initiatives and in the development of process-aware information systems. The process models resulting from such projects have to meet high quality standards since the costs of correcting errors grow exponentially if they are discovered at later stages. Nevertheless, it has been observed that process models in practice have a substantial amount of quality issues [Men08, Men09], which can become a threat for the success of the whole modeling project [BGR05].

The reasons for bad quality can be manifold. One of the central issue in this context is the fact that the persons involved with a modeling project might not have sufficient experience with modeling [Ros06]. This can be the case when business professionals are asked to model the processes of their department themselves or when process analysts have to interact with business professionals who are not capable or willing to study the created process models. A way to deal with these challenges is to enhance the modeling process with steps for validation and verification. But while verification has been intensively studied in recent research [vdAvHtH$^+$11], there has been a notable research gap on how the validation of a process model can be supported by a modeling tool if the modeler is not a modeling expert.

In this paper we address the challenge of helping non-expert modelers in validating process models. We define an approach to generate information-equivalent natural language from a process model which can be read and understood by persons without specific modeling skills. As a proof of concept, we have implemented the required functionality and integrated it into the Petri net modeling tool WoPeD. Our implementation builds on insights from multimedia learning theory which emphasizes the advantages of having a complementary text to a graphical representation [May09]. This design contribution informs research on process validation and provides the basis for evaluative research in this area.

The remainder of this paper is structured as follows. Section 2 describes the background of our research, that is the complementary roles involved in a modeling projects, the phases of modeling, and the theoretical principles upon which tool support can be defined. Section 3 defines the different stages of our text generation technique. We utilize concepts from natural language processing and process model parsing in order to serialize the content of a model as text. Section 4 presents our prototypical implementation of validation support in the process modeling tool WoPeD. Section 5 discusses our approach in the light of related work before Section 6 concludes the paper and gives an outlook on future research.

2 Background

In this section we discuss the modeling lifecycle and how model validation is challenged by differences in modeling expertise between system analyst and domain expert. Then, we discuss in how far a textual representation of a process model can help in model validation.

2.1 Validation in the Modeling Lifecycle

The classical setting of system analysis and design is defined as an interplay between two distinct roles: the domain expert and the system analyst [FW06]. The interaction between these two roles can be described in terms of a systems modeling lifecycle including four stages. First, the system analyst interviews the domain expert in order to *elicit* process-relevant information. Second, the system analyst then organizes this information by *modeling* parts of the process. Third, the system analyst conducts a *verification* step on the model with the aim to resolve inconsistencies. Fourth, the system analyst show the model to the domain expert, explains the content and tries to get feedback in order to *validate* the model. The domain expert is typically familiar with the details of the process, but has limited understanding of business process modeling techniques. The system analyst in contrast has elaborate methodological skills, but has to rely on the information provided by the domain expert on how the business process works in practice.

As an example, consider a process modeling project in a hotel. The system analyst is asked to specify the hotel service process, and therefore has to talk to the waiter, the kitchen personnel, the sommelier, and the room-service manager. The result of these interviews is a Petri Net capturing the hotel service process. It deals with an incoming order which

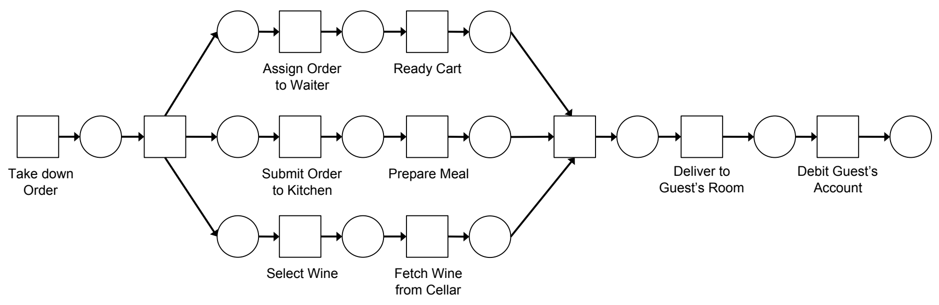


Figure 1: Example Petri Net Capturing a Hotel Service Process

is triggering three concurrent streams of action: The waiter readies the cart, the kitchen prepares the meal and the wine is selected and fetched from the cellar. Once these tasks are finished, the order is delivered and the guest's account is debited. In order receive meaningful feedback on this model, the system analyst consults each of these parties, shows them the model and explains what is happening. The domain experts can then provide feedback towards improving and correcting the model.

The challenges towards getting meaningful feedback from domain experts is twofold. The first issue relates to *familiarity*. Since domain experts typically lack familiarity with process modeling and formal notations, they are likely to refuse working with such diagrams. Therefore, missing familiarity often leads to lack of acceptance and the demand to discuss the process in a natural-language way. Second, even if domain experts are willing to accept process models, they often do not have the required *expertise* to discuss and analyze a model in detail. The consequence in such a case is that they feel overwhelmed by syntactic aspects of the model, blocking their cognitive capacities to investigate semantical and pragmatic aspects of the model.

2.2 Natural Language Text as a Basis for Model Validation

In this section, we aim to investigate in how far a textual representation of the contents of a process model can help a domain expert to understand the content of a model. There is a rich body of research on the pros and cons of visual diagrams in contrast to natural language. The hypothesis by Larkin and Simon that a diagram is sometimes worth ten thousand words [LS87] has inspired this stream of research. Their observation is that text is limited to linear order while the spatial arrangement of different elements in a diagram allows for a more efficient information processing. However, symbols of a graphical notations have to be learnt by human readers in order to be understood [Sia04]. This fact is exactly what sets system analysts and domain experts aside in terms of their model readership skills. The empirical findings on the strengths and weaknesses of text and diagrams are diverging. Moher *et al* [MMBL93] looked at several ways to express program structures in text and in a Petri Nets, and state "for our tasks, graphics were no better than text, and in several cases were considerably worse". A comprehensive summary of experimental results is provided in [Whi97]. Several of these studies suggest that expertise is the most relevant factor in comprehension, e.g. [CSKB$^+$89], but there is no absolute better or worse representation. Therefore, recent research suggests providing models with complementary text [OFR$^+$12].

Different aspects have to be considered for the joint presentation of text and model. The theory of multimedia learning proposed by Mayer [May09] helps to tailor the interaction between a learner and a medium in an efficient and effective manner. In the context of the validation situation, the domain expert can be understood as a learner who aims to comprehend a process model in order to provide feedback about its correctness and completeness. Our focus in this context is to help understanding by *providing additional natural language text*. For such a situation, several principles of multimedia learning can be considered towards improving understanding and learning. We discuss a subset of Mayer's principles which are most relevant.

- The *multimedia principle* states that people learn better when a figure and a text than from text alone. The principle is routed in the theoretical consideration that learners can construct complementary mental models from verbal and visual information, and link them together.
- The *signaling principle* argues that people comprehend material more effectively when cues are added that help in understanding the organization of content. This aspect is relevant when process models become large. On the one hand, important aspects need to be highlighted while content of less relevance is hidden.
- The *spatial continguity principle* recommends presenting corresponding text and figures near to each other. This implies that a correspondence between parts of the visual model and the textual description should be visible.
- The *segmenting principle* suggests that multimedia material can be better understood if the user can progress in a self-paced way. This principle rules out several design options that would show text in a powerpoint-like animated fashion.
- The *expertise reversal effect* states that cues which are helpful to unexperienced learns might be disturbing to experts [KACS03]. This means all tool functionality should be designed in such a way that domain experts benefit while experts are not distracted.

The summarized theoretical considerations potentially help to organize the interaction with a process model in a way that helps to understand the domain more deeply. Accordingly, we hypothesize that the provision of textual information about a process is likely to help domain experts to understand a process model. The research objective of this work is to leverage these multimedia principles and to investigate how textual information can be automatically provided if the process model is available. We aim to discuss the feasibility of our approach based on a prototypical implementation.

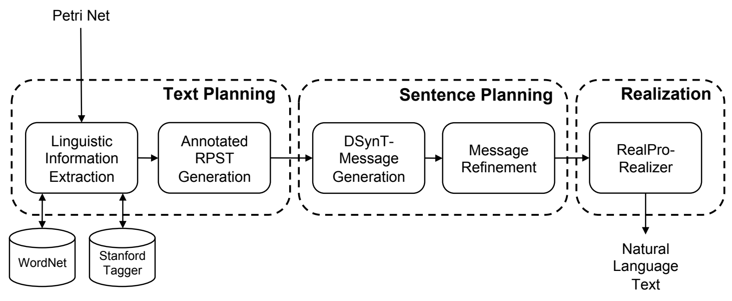


Figure 2: Architecture of the Natural Language Generation System

3 Generating Text from Petri Nets

In this section we introduce the approach for generating natural language texts from Petri Nets. It is based on an approach we defined in prior work [LMP12]. Figure 2 gives an overview of the architecture which is consisting of five main components building a pipeline. In the following subsections we will briefly introduce each component.

3.1 Linguistic Information Extraction

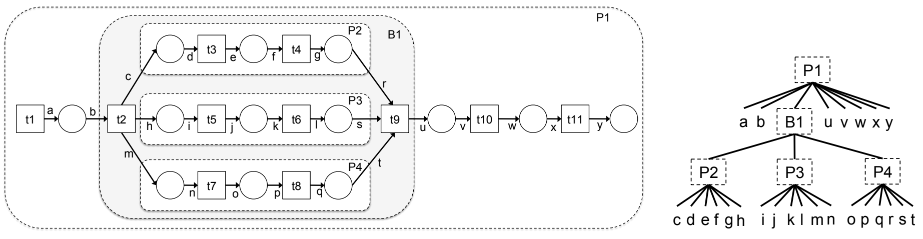
Goal of this component is the adequate inference of the already available linguistic information from the Petri Net. Therefore, we employ an automatic annotation technique we defined in prior work [LSM12]. In this technique we use the Stanford Parser [KM03] and the lexical database WordNet [Mil95] to recognize different label patterns which we identified in the context of an extensive analysis of industry proces model collections. In this way we can reliably extract action, business object and the additional information fragment from each labeled transition in the Petri Net.

As an example consider the transition *Fetch Wine from Cellar* from Figure 1. Using Wordnet we can determine that only the first word in the label can be a verb. Being aware of regular labeling structures, we hence can conclude that we face a label following the verb-object style. This style is characterized by an imperative verb which is followed by the business object and the additional information fragment. As a result, we can derive the action *fetch*, the business object *wine* and the additional information fragment *from cellar*. Once this extraction has been performed for all labeled transitions, the derived annotation records are passed to the next component.

3.2 Annotated RPST Generation

The RPST Generation module derives a tree representation from the given Petri Net in order to provide a basis for a step by step process description. In particular, we compute a Refined Process Structure Tree (RPST) which is a parse tree containing a hierarchy of subgraphs derived from the Petri Net [VVK09, PVV11]. The computation of the RPST is based on the observation that every workflow graph can be decomposed into a hierarchy of logically independent subgraphs having a single entry and single exit. Thereby, any two of these subgraphs, which are also called fragments, are either nested or disjoint. The resulting hierarchy can be shown as a tree where the root captures the entire tree and the leaves are fragments with a single arc.

In total an RPST may contain four different fragment types: trivial fragments (T), bonds (B), polygons (P) and rigids (R). Trivial fragments consist of two nodes connected with a single arc. A bond represents a set of fragments sharing two common nodes. In Petri Net this applies for transition-place constructions which are representing splits and joins. Polygons are capturing sequences of other fragments. If a fragment cannot be assigned to one of the latter classes, it is categorized as a rigid. Figure 3 illustrates the concepts using an abstracted version of the hotel process and its corresponding RPST.



(a) Abstract Version of Example Process (b) RPST

Figure 3: Abstract Version of Figure 1 and its RPST

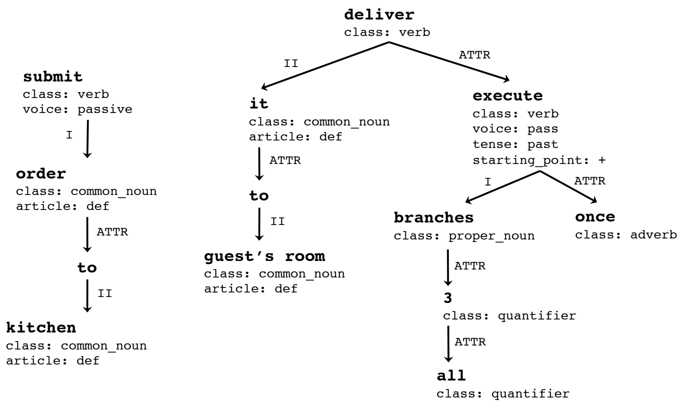
After we have derived the RPST, we automatically annotate it with the linguistic information from the extraction phase and with additional meta information. For instance, the transition *t1* from the trivial fragment *a* is annotated with the action *take down* and the business object *order*. The bond *B1* is tagged as an AND-split as we face a single transition with three outgoing places.

3.3 DSynT-Message Generation

The message generation component maps the annotated RPST elements to a list of intermediate messages. More specifically, each sentence is stored as a deep-syntactic tree (DSynT), which is a dependency representation introduced by the Meaning Text Theory [MP87]. In a deep-syntactic tree each node reflects a component of the represented sentence and is labeled with the according lexeme. In addition each node carries grammatical meta information as for instance the voice and tense of verbs or the number and definiteness of nouns. The advantages deep-syntactic trees are the rich but manageable representation of sentences and the existence of off-the-shelf surface realizers which automatically map deep-syntactic representations to grammatically correct sentences.

By recursively traversing the RPST, we generate DSynT-based messages for trivial fragments and bonds. Using the annotation from the RPST, each transition can be directly mapped to a DSynT. For illustrating this procedure consider the transition *Submit Order to Kitchen* from the example process (Figure 1). This transition contains the action *submit*, the business object *order* and the additional information fragment *to kitchen*. Having this annotation information at hand, we can easily construct a DSynT representing the sentence *The order is submitted to the kitchen*. Note that the sentence is presented in passive voice as Petri Nets do not contain any information about involved roles. Figure 4(a) illustrates the derived DSynT graphically.

For the transformation of bonds we use predefined (but editable) DSynT-templates in order to express the upcoming split or join. As an example consider the AND join from the example model. The conditional clause *Once all three branches were executed* is passed to the first transition after the bond (*Deliver to Guest's Room*) and incorporated accordingly. Figure 4(b) visualizes the according DSynT.



(a) Simple DSynT (b) DSynT with Conditional Sentence

Figure 4: Deep-Syntactic Tree of Two Messages from Figure 1

3.4 Message Refinement

Within the message refinement component, we take care of message aggregation, referring expression generation and discourse marker insertion.

The need for *message aggregation* arises when the considered process contains long sequences. In such cases we aggregate messages which share the same business object or action. Thus for instance two separate sentences referring to the same action are merged to a single sentence.

If there are still adjacent messages with the same business object, it is replaced with a *referring expression*. We use WordNet for replacing a business object with a suitable pronoun. More specifically, we derive the hypernyms of the considered business object. If we for example look up the business object *guest*, we will find the hypernym *person* which is indicating that this business object must be replaced by *he* or *she*. By contrast, the hypernyms of *order* only contain entries like *artifact* or *entity*. Hence, the business object *order* is referenced with *it*.

For the *discourse marker insertion* we use an extendible set of connectors, to insert markers such as *then* and *afterwards*. In this way, we obtain a well readable text with sufficient variety.

3.5 Surface Realization

The complexity of the surface realization task has led to the development of publicly available realizers. Thereby, these tools significantly vary in aspects such as license costs, generation speed and Java compatibility. Taking these aspects into account, we decided to use the DSynT-based realizer RealPro from CoGenTex [LR97]. RealPro requires an XML-based DSynT message as input and transforms it to a grammatically correct sentence.

4 Implementation

In this section, we present details of our prototypical implementation. We first describe the interface of the text generation service. Then, the integration of the concepts into WoPeD is illustrated. Finally, we revisit principles from the theory of multimedia learning and discuss how they have been addressed.

4.1 Text Generation Web Service

In order to make the text generation functionality available, we created a web service[1]. Thereby, the web service directly builds on the Java based implementation of the text generation technique. We designed a service interface defining a Petri Net in the XML based format PNML as input. As a result, the web service returns a PNML file with a newly created element $text$. This element contains a list of $phrase$ elements linking each generated sentence to one or more transitions. Listing 1 shows an excerpt from the returned PNML file illustrating how the links between text and Petri Net are established.

[1]The web service can be accessed via http://ai.wu.ac.at/ProcessToTextService/ProcessToText

Listing 1: Excerpt from returned PNML file

```
<text>
<phrase ids="t3"> The process begins when an order is taken down. </phrase>
<phrase ids="t1"> Then, the process is split into 3 parallel branches. </phrase>
<phrase ids="t7"> The wine is selected. </phrase>
<phrase ids="t11"> Afterwards, the wine is fetched from the cellar. </phrase>
<phrase ids="t2"> The order is assigned to the waiter. </phrase>
<phrase ids="t4"> Subsequently, the cart is readied. </phrase>
<phrase ids="t5"> The order is submitted to the kitchen. </phrase>
<phrase ids="t9"> Then, the meal is prepared. </phrase>
<phrase ids="t6,t1"> Once all 3 branches were executed it is delivered to the  guest'
   s room. </phrase>
<phrase ids="t14"> Afterwards, the guest's account is debited. </phrase>
</text>
```

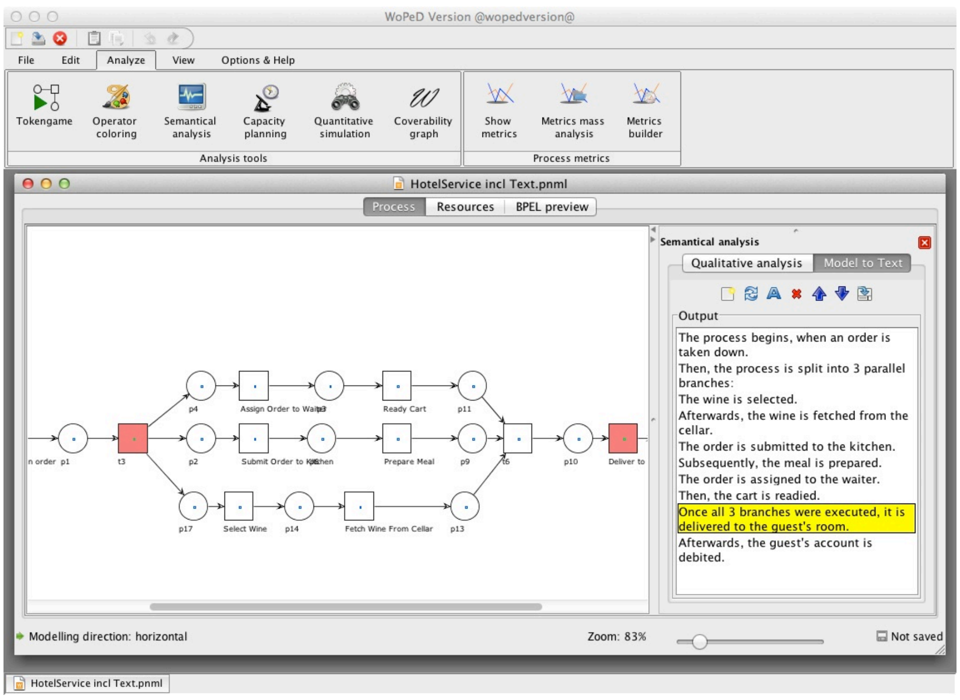
4.2 Tool Integration

In order to demonstrate how the introduced algorithm can support the validation of process models we integrated it into the modeling tool WoPeD. We chose WoPeD as it is an open source process modeling tool for Petri Nets which is completely implemented in Java and can hence be accordingly extended. It was developed at the Baden-Wuerttemberg Cooperative State University (DHBW) and is available for different platforms such as Windows, Mac or Linux. To provide the user with the text generation functionality, we added a new window to the user interface. It is positioned on the right hand side of the modeling window and can be flexibly displayed and hidden. If the user wishes to generate a textual description for the given process model, he can chose between two options. First, a textual description can be automatically generated by the previously introduced web service. Alternatively, the user can manually create or adapt the description. Figure 5 shows a screenshot from the extended WoPeD with an exemplary Petri Net and the corresponding text generated by our technique. In order to clearly show the associations between the Petri Net elements and the generated sentences, the according Petri Net elements are highlighted if the user clicks on a sentence. By double clicking on a sentence, the user can manually adapt the generated text or add additional information.

To enable the user to save the generated texts, we implemented two different options. First, the text can be saved to the XML-based format PNML. Thereby, it is saved under the new element *text* as described in Section 4.1. Second, the generated text can be exported as plain text file. This file only contains the description and, if required, the IDs to the Petri Net elements. As a result, the text can also be distributed to persons who are not working with WoPeD and can hence support the validation of the underlying Petri Net.

4.3 Incorporation of the Multimedia Principles

In Section 2 we discussed the multimedia principles introduced by Mayer and how the effect the learning process of humans. Being aware of these insights, our tool accordingly



The process begins, when an order is taken down.
Then, the process is split into 3 parallel branches:
The wine is selected.
Afterwards, the wine is fetched from the cellar.
The order is submitted to the kitchen.
Subsequently, the meal is prepared.
The order is assigned to the waiter.
Then, the cart is readied.
Once all 3 branches were executed, it is delivered to the guest's room.
Afterwards, the guest's account is debited.

Figure 5: User Interface of Woped with additional text frame

supports some essential multimedia principles in order to foster effective learning in the context of process models. The following principles are reflected by our tool:

- *Multimedia Principle*: The basic idea of generating texts for process models is based on the multimedia principle. As the text and the model are processed by different channels, the introduced tool fosters the learning, i.e. the understanding of process models.
- *Signalizing Principle*: For bigger and complex process models the resulting text may quickly become huge such that it is hard for the reader to associate a sentence with the according element in the net. Hence, we implemented the signalizing principle by highlighting the appropriate sentence if the user clicks on an element in the model. Accordingly, the corresponding element is highlighted if a sentences is selected in the list. As this functionality reduces the extraneous processing it enables the user to focus on the essential information in the model.
- *Spatial Contiguity Principle*: Due to the closeness of the text and the model in the editor, we consider the spatial contiguity principle to be fulfilled. This is important as the association of distant elements causes cognitive load preventing the user to focus on the essential information.

- *Segmenting Principle*: Huge models may quickly overwhelm the user with too much information. Accordingly, we implemented the functionality to save text for sub processes. In this way, the process can be decomposed into manageable units without refraining from creating a textual description.
- *Expertise reversal effect*: In order to reflect the different levels of knowledge of the target groups, we implemented the possibility to edit the generated texts. In this way, redundancy between model and text can be explicitly implemented or removed. If, for example, the target group has only little experience with process models, the generated descriptions can be complemented with additional information. Otherwise, extensive generated descriptions can be shortened to the essential message.

The mentioned principles helped to rule out design options. The principles also inspired the introduction of functionality. The next steps will be to conduct experiments with novice and expert process modelers in order to investigate to which degree the provided text leverages understanding, at least in the novices cohort. A second important treatment will be the variation of the modeling tasks in terms of complexity. It will be interesting to study how much text can support the understanding of large models.

5 Related Work

The work presented in this paper can be related to three major streams of research: natural language generation systems, process model comprehension and process model validation.

The generation of natural language texts has a long tradition and has been applied in various scenarios. Examples include the generation of weather forecasts [GDK94] or the documentation of the activities of planning engineers [MKS94]. The application of text generation techniques on conceptual models is limited to a few examples. The ModelExplainer generates natural language descriptions from object models [BL96] and the GeNLangUML creates textual specifications from UML class diagrams [MAA08]. However, none of these approaches tackles the specific problems associated with process models. Accordingly, our approach complements these techniques by introducing a strategy to appropriately verbalize control-flow-based aspects.

The field of process model comprehension is discussed from different perspectives. For instance, the results from [MRC07] show that the number of arcs has an important effect on the overall model understandability. The work in [MRR10] demonstrates the impact of the natural language in the activity labels for model comprehension. A more general perspective is taken in [ZPW11], where the authors investigate in how far the cognitive inference process affects the model understanding. The approach presented in this paper builds on these insights as it tries to lower the overall burden of process model comprehension. If a user is is not able to interpret a given model due to confusing wording or an overwhelming number of arcs, the generated text can guide the user through the model.

The role and the need of natural language feedback in the context of model validation is addressed in [Dal92, RP92]. They propose the generation of natural language text as an appropriate solution. This is also in line with the approach of ORM and NIAM in terms of verbalization [VvB82, NH89]. The communication problem between domain expert and system analyst has also been recognized for declarative models. For instance in [ZPW12] this problem is addressed by Test Driven Modeling. The advantages of combination of text and diagram have been intensively debated in the area of visual programming languages (see [Whi97] for an overview). Nowadays, most authors agree on the fact that text plus diagram provides better comprehension than any of the two in isolation [OFR$^+$12, May09]. Hence, our technique may ease the validation of process models. Using the introduced tool, the domain expert is guided through the process step by step and can more easily decide about its semantic correctness.

6 Conclusion

In this paper, we presented an automatic approach for generating natural language texts from Petri Nets. The approach builds on the insights from multimedia leaning theory and combines natural language analysis, graph decomposition techniques and a linguistic framework for the generation of comprehensive texts. As a proof of concept we implemented our technique as web service and incorporated it into the open source modeling tool WoPeD. By building on the principles of Mayer's multimedia theory the tool provides the user with an easy and supportive interface. A graphical highlighting mechanism illustrates the links between sentences and model elements. An edit and export function enables the user to create and distribute target group specific texts. As a result, our technique is a first step towards a tool-supported validation of process models.

In future research we aim for demonstrating the potential of our tool to ease the validation of process models and improve the communication between domain expert and modeler. Consequently, we plan to conduct an industry case where we can gather qualitative feedback and can accordingly improve our tool. In addition, we plan an experiment which shows that the detection of semantic deficiencies in a model is easier when the user is provided with a combination of model and text. Finally, we aim to explore how we can also address the auditory channel. Having generated a descriptive text for a given process model, this text could also be read out to the user. Such a technique would nicely complement the approach introduced in this paper.

References

[BGR05] W. Bandara, G. G. Gable, and M. Rosemann. Factors and Measures of Business Process Modelling: Model Building through a Multiple Case Study. *European Journal of Information Systems*, 14:347–360, 2005.

[BL96] Ehud Reiter Benoit Lavoie, Owen Rambow. The ModelExplainer. In *Proceedings of the 8th international workshop on natural language generation*, pages 9–12, 1996.

[CSKB+89] B. Curtis, S.B. Sheppard, E. Kruesi-Bailey, J. Bailey, and D. Boehm-Davis. Experimental evaluation of software documentation formats. *Journal of Systems and Software*, 9(2):167–207, 1989.

[Dal92] H Dalianis. A method for validating a conceptual model by natural language discourse generation. *Advanced Information Systems Engineering*, pages 425–444, 1992.

[FW06] P.J.M. Frederiks and T.P. van der Weide. Information Modeling: The Process and the Required Competencies of Its Participants. *Data & Knowledge Engineering*, 58(1):4–20, 2006.

[GDK94] E. Goldberg, N. Driedger, and R.I. Kittredge. Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9(2):45–53, 1994.

[KACS03] S. Kalyuga, P. Ayres, P. Chandler, and J. Sweller. The expertise reversal effect. *Educational Psychologist*, 38(1):23–31, 2003.

[KM03] D. Klein and C. D. Manning. Fast Exact Inference with a Factored Model for Natural Language Parsing. In *NIPS 2003*, volume 15. MIT Press, 2003.

[LMP12] H. Leopold, J. Mendling, and A. Polyvyanyy. Generating Natural Language Texts from Business Process Models. In *24th International Conference on Advanced Information Systems Engineering (CAISE 2012)*, Gdansk, Poland, 2012.

[LR97] B. Lavoie and O. Rambow. A fast and portable realizer for text generation systems. In *Applied natural language processing*, pages 265–268. ACL, 1997.

[LS87] J.H. Larkin and H.A. Simon. Why a Diagram is (Sometimes) Worth Ten Thousand Words**. *Cognitive science*, 11(1):65–100, 1987.

[LSM12] Henrik Leopold, Sergey Smirnov, and Jan Mendling. On the refactoring of activity labels in business process models. *Information Systems*, 37(5):443 – 459, 2012.

[MAA08] Farid Meziane, Nikos Athanasakis, and Sophia Ananiadou. Generating Natural Language specifications from UML class diagrams. *Requirements Engineering*, 13:1–18, 2008.

[May09] R. E. Mayer. *Multimedia Learning*. Cambridge University Press, second edition edition, 2009.

[Men08] J. Mendling. *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*, volume 6 of *LNBIP*. Springer, 2008.

[Men09] J. Mendling. Empirical Studies in Process Model Verification. *LNCS Transactions on Petri Nets and Other Models of Concurrency II, Special Issue on Concurrency in Process-Aware Information Systems*, 2:208–224, 2009.

[Mil95] G. A. Miller. WordNet: a Lexical Database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[MKS94] K. McKeown, K. Kukich, and J. Shaw. Practical issues in automatic documentation generation. In *Applied natural language processing*, pages 7–14. ACL, 1994.

[MMBL93] T. G. Moher, D. C. Mak, B. Blumenthal, and L. M. Leventhal. Comparing the comprehensibility of textual and graphical programs: the case of Petri nets. In *Empirical Studies of Programmers: Fifth Workshop*, pages 137–161. Norwood, NJ: Ablex., 1993.

[MP87] Igor Mel'cuk and Alain Polguère. A Formal Lexicon in the Meaning-Text Theory (or How to Do Lexica with Words). *Computational Linguistics*, 13(3-4):261–275, 1987.

[MRC07] Jan Mendling, Hajo Reijers, and Jorge Cardoso. What Makes Process Models Understandable? In Gustavo Alonso, Peter Dadam, and Michael Rosemann, editors, *Business Process Management*, volume 4714 of *Lecture Notes in Computer Science*, pages 48–63. Springer Berlin / Heidelberg, 2007.

[MRR10] J. Mendling, H. A. Reijers, and J. Recker. Activity Labeling in Process Modeling: Empirical Insights and Recommendations. *Information Systems*, 35(4):467–482, 2010.

[NH89] G.M. Nijssen and T.A. Halpin. *Conceptual Schema and Relational Database Design: a fact oriented approach.* Prentice-Hall, Inc., 1989.

[OFR$^+$12] Avner Ottensooser, Alan Fekete, Hajo A. Reijers, Jan Mendling, and Con Menictas. Making sense of business process descriptions: An experimental comparison of graphical and textual notations. *Journal of Systems and Software*, 85(3):596–606, 2012.

[PVV11] A. Polyvyanyy, J. Vanhatalo, and H. Völzer. Simplified Computation and Generalization of the Refined Process Structure Tree. In *Web Services and Formal Methods*, volume 6551 of *LNCS*, pages 25–41. Springer, 2011.

[Ros06] M. Rosemann. Potential Pitfalls of Process Modeling: Part A. *Business Process Management Journal*, 12(2):249–254, 2006.

[RP92] C. Rolland and C. Proix. A natural language approach for Requirements Engineering. In *Advanced Information Systems Engineering*, volume 593 of *LNCS*, pages 257–277. Springer, 1992.

[Sia04] K. Siau. Informational and computational equivalence in comparing information modeling methods. *Journal of Database Management*, 15(1):73–86, Jan.-Mar 2004.

[vdAvHtH$^+$11] Wil M. P. van der Aalst, Kees M. van Hee, Arthur H. M. ter Hofstede, Natalia Sidorova, H. M. W. Verbeek, Marc Voorhoeve, and Moe Thandar Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Asp. Comput.*, 23(3):333–363, 2011.

[VvB82] G.M.A. Verheijen and J. van Bekkum. NIAM, aN Information Analysis Method. In *Proc. of the IFIP WG8.1 Working Conference on Comparative Review of Information System Methodologies*, pages 537–590. North-Holland, 1982.

[VVK09] Jussi Vanhatalo, Hagen Völzer, and Jana Koehler. The refined process structure tree. *Data & Knowledge Engineering*, 68(9):793 – 818, 2009.

[Whi97] K.N. Whitley. Visual programming languages and the empirical evidence for and against. *Journal of Visual Languages and Computing*, 8(1):109–142, 1997.

[ZPW11] Stefan Zugal, Jakob Pinggera, and Barbara Weber. Assessing Process Models with Cognitive Psychology. In Markus Nüttgens, Oliver Thomas, and Barbara Weber, editors, *EMISA*, volume 190 of *LNI*, pages 177–182. GI, 2011.

[ZPW12] Stefan Zugal, Jakob Pinggera, and Barbara Weber. Toward enhanced life-cycle support for declarative processes. *Journal of Software: Evolution and Process*, 24(3):285–302, 2012.