Transforming Existing Procedural Business Processes into a Constraint-Based Formalism

Floor Vermeer
Be Informed
Apeldoorn, the Netherlands
Email: f.vermeer@beinformed.com

Jeroen van Grondelle
Be Informed
Apeldoorn, the Netherlands
Email: j.vangrondelle@beinformed.com

Eline de Haan
University of Applied Sciences Utrecht
Utrecht, the Netherlands
Email: eline.dehaan@student.hu.nl

Slinger Jansen
Utrecht University
Utrecht, the Netherlands
Email: Slinger.Jansen@uu.nl

Martijn Zoet
University of Applied Sciences Utrecht
Utrecht, the Netherlands
Email: martijn.zoet@hu.nl

Abstract

Many organizations use business process management to manage and model their processes. Currently, flow-based process formalisms, such as BPMN, are considered the standard for modeling processes. However, recent literature describes several limitations of this type of formalism that can be solved by adopting a constraint-based formalism. To preserve economic investments in existing process models, transformation activities needed to be limited. This paper presents a methodical approach for performing the tedious parts of process model transformation. Executing the method results in correctly transformed process models and reduces the effort required for converting the process models.

Keywords

Business Process Management, Process model transformation, Declarative, Procedural, Method Engineering

INTRODUCTION

Many organizations use business process management to manage and model their processes. Flow-based formalisms such as BPMN are considered the current standard for business process modeling (Mendling, Recker, & Reijers, 2011; Recker, 2010; Zur Muehlen & Recker, 2008). Consequently, a large amount of time and effort has already been spent by organizations on modeling their processes with a flow-based formalism.

However, modeling business processes with a flow-based formalism results in several challenges (van der Aalst, Weske, & Grünbauer, 2005; Van Grondelle & Gulpers, 2011). First, the model is an interpretation within business constraints, the constraints itself are not modeled. Consequently, traceability of concepts and order of concepts to original business constraints is often lost (Almeida, Eck, & Iacob, 2006). When changes to the process model are required, it is unclear if these changes violate the original constraints. Furthermore, the context of the process is not taken into account and it creates problems dealing with complex and changing environments (van der Aalst et al., 2005; Van Grondelle & Gulpers, 2011).

Constraint-based formalisms have been proposed to counter these challenges (Goedertier, Haesen, & Vanthienen, 2007; Pesic & van der Aalst, 2006; van der Aalst et al., 2005; Van Grondelle & Gulpers, 2011). In contrast to

procedural process models, declarative process models contain explicit business constraints from which the execution scenario is inferred (Goedertier & Vanthienen, 2007; Van Grondelle, Zoet, & Vermeer, 2013). Since business constraints are modeled explicitly, traceability of these constraints is ensured. Furthermore, the order of concepts is determined based on the context of an instance and the relevant business constraints. This alleviates the problem of context tunneling and allows for flexibility at runtime.

Adoption of a constraint-based seems apparent. However, completely obliterating current investments, as was the notion in the 90's might be too large of a barrier for organizations to adopt this type of formalism (Davenport & Stoddard, 1994; Hammer, 1990).

The goal of this paper is to develop a methodical way of capturing the original business constraints from existing procedural process models in a constraint-based formalism. Concepts in a procedural process model are modeled within business constraints, but the concepts and order of concepts in that model can often not be traced back to their original business constraints (Almeida et al., 2006). Consequently, when a procedural model is transformed to a constraint-based formalism, the transformed model still represents an implementation within the business constraints instead of the business constraints itself. A direct transformation from procedural to declarative models does not provide enough merit, since the limitations of the procedural model are then included in the declarative model (Figure 1, A and B). To make to declarative model represent the original business constraints more closely, interpretation of the transformed process model is required (Figure 1, C). By interpreting the process model, original business constraints are explicated and can be used to enrich the declarative process model. Based on these considerations the following research question is formulated: *"How can formal underlying business constraints be extracted from procedural business process models?*"

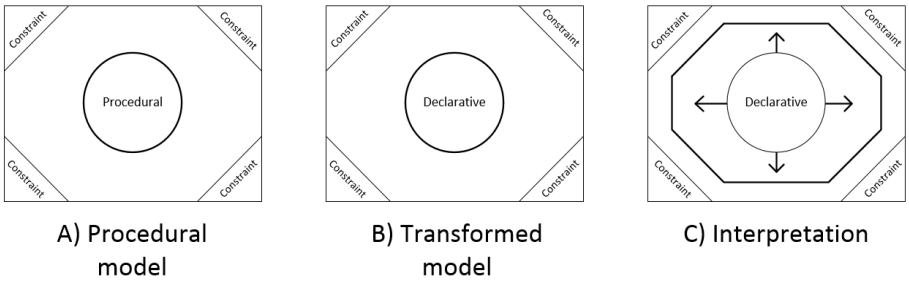

Figure 1: Transformation steps. Adapted from Pesic & Van der Aalst (2006).

To perform the transformation and interpretation on procedural process models, a method is created using the method engineering approach (Brinkkemper, 1996; van de Weerd & Brinkkemper, 2008). To determine the content of the method, techniques are selected which can be used to transform process models or provide an interpretation of process model concepts. The method is presented in a Product Deliverable Diagram (PDD), as described in Van de Weerd & Brinkkemper (2008). The method will be used to: 1) identify constraints; and 2) capture these constraints in a constraint-based formalism. Results from performing the method will be a model in a constraint-based formalism based on a transformation and interpretation of a procedural model (Figure 2).

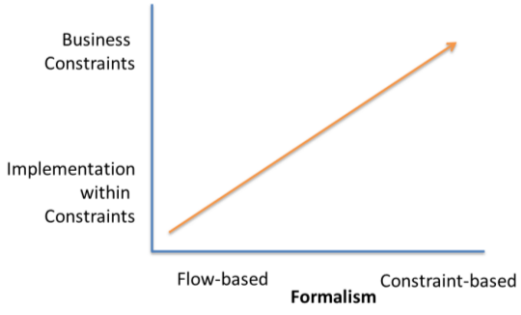

Figure 2: Transformation

In the next section several flow-based and constraint-based business process formalisms are described. Furthermore, techniques that can be used to transform and interpret process models are explained.

RELATED LITERATURE

For the purpose of modeling business processes, two different types of modeling formalisms can be distinguished (Goedertier & Vanthienen, 2007): Flow-based formalisms and Constraint-based formalisms. Resulting in procedural process models and in declarative process models respectively (Goedertier & Vanthienen, 2007).

Process models are considered procedural when the focus is on the order of activities. The execution scenario is explicit and designed within implicit business constraints (Goedertier & Vanthienen, 2007).

Typical examples of flow-based process formalisms are Petri Net, YAWL and BPMN (OMG, 2011; Peterson, 1981; van der Aalst & ter Hofstede, 2005). Wohed et al. (2006) provide an overview of the functionalities of more recent formalisms. According to this overview, BPMN currently offers most functionality when modeling workflow patterns. This has been confirmed by research which compared different formalisms using the Bunge-Wand-Weber framework (Recker, Indulska, Rosemann, & Green, 2005).

Several limitations of procedural models have been described in recent literature. Van der Aalst et al. (2005) mention four problems with flow-based formalisms: First, there is a discrepancy in modeling atomic activities and the real world activities they represent. Secondly, distribution and authorization of work packages is often combined, which sometimes require tedious workarounds to solve issues caused by this combination. Thirdly, by focusing on the sequence of activities the context of the organization is not taken into account. This can lead to errors and inefficiencies. Last, traditional flow-based formalisms prescribe what should be done, instead of what can be done. Other research describes a fifth problem associated with flow-based formalisms. Pesic & van der Aalst (2006) and van Grondelle & Gulpers (2011) state that flow-based formalisms are inefficient in dealing with rapidly changing and complex business environments. In traditional workflow management systems, making changes can be time consuming and complex. When the original business constraints are modeled implicitly, information regarding specific design choices is lost and traceability of these constraints becomes obscure (Almeida et al., 2006; Van Grondelle & Gulpers, 2011). Any change to the model can then become an unintentional violation of the original business constraints. Furthermore, dealing with a dynamic environment requires a large number of process variants, which overcomplicates the process models (Van Grondelle & Gulpers, 2011).

Multiple constraint-based process modeling formalisms have been developed recently. Condec is a constraint-based formalism with transitions connected to each other with linear temporal logic (LTL) constraints (Pesic & van der Aalst, 2006). This type of constraints allows for inferring the order of concepts, based on the state of a specific instance. Another declarative formalism is EM-BRA²CE, which is based on Semantics of Business Vocabulary and Rules (SBVR) (Goedertier et al., 2007). It has a textual syntax which supports 16 different types of business constraints, such as availability pre-conditions or an activity post-condition. The formalism infers which activities to perform based on constraints and the goal of the process. Lastly, the Declarative Process Modeling Notation (DPMN) is based on pre- and post-conditions and goal-orientation (Van Grondelle & Gulpers, 2011). It offers a visual syntax to specify pre- and post-conditions between activities, artifacts, roles and other concepts (Figure 3). The formalism infers which activities need to be performed and which concepts need to be available based on the constraints and goal of the process.

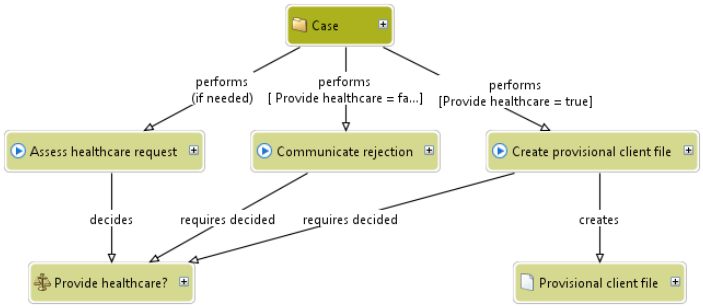

Figure 3: Example of a DPMN process model

Transformation techniques

A distinction is made between techniques that perform model transformation and model interpretation. Graph transformation is a transformation technique which has its origins in mathematics (H. Ehrig, Prange, & Taentzer, 2004). A graph transformation consists of meta-models and productions to transform concepts from a source formalism to a target formalism. In this paper graph transformation is used to transform single concepts, such as a sequence flow and activities from a procedural model into a declarative formalism. Dijkman, Dumas and Ouyang (2008) specify Well-formed BPMN structures which are used to transform BPMN into Petri net. By also looking at the context of certain concepts, Well-formed BPMN transforms clusters of BPMN concepts more precisely. In the context of this paper, Well-Formed BPMN can be used to transform pre-defined structures from a procedural model into a declarative formalism.

The related cluster pair technique can be used to retrieve specific clusters based on a similarity measurement (Niemann, Siebenhaar, Schulte, & Steinmetz, 2012). The similarity measurements are performed

on activity labels and are based on syntactic and semantic similarity. To transform a procedural model into a declarative formalism, predefined clusters can be used as queries. When a cluster is determined to be similar, the cluster is replaced by a corresponding cluster in the declarative formalism. The semantic process similarity technique is similar to the related cluster pair technique (M. Ehrig, Koschmider, & Oberweis, 2007). Instead of looking for similar clusters, the semantic process similarity technique can be used to find process models that are similar. A complete procedural process model can be replaced with a declarative process model when it is determined similar to the process model used as a query.

Lexical analysis of activity labels can be used to classify activities in the procedural model based on the verbs in their label. Classification schemes, such as described in Mendling et al. (2011) can be used to perform the classification. Based on the classification, an activity can be changed, enriched or omitted. A Naïve Bayes classifier can also be used to classify activity labels (Tan, Steinbach, & Kumar, 2006). The words used in activity labels serve as attributes on which the classification is based.

RESEARCH METHOD

The artifact resulting from this research is a method. Since an artifact are created during this research, using a design research approach is justified (March & Smith, 1995). Hevner, Ram, March and Park (2004) provide a conceptual framework and guidelines which are used to construct the design research project presented in this paper. Furthermore, research provided by Peffers, Tuunanen, Rothenberger and Chatterjee (2007) and Verschuren and Doorewaard (2007) were used to guide the research project. The framework can be viewed in Figure 4.

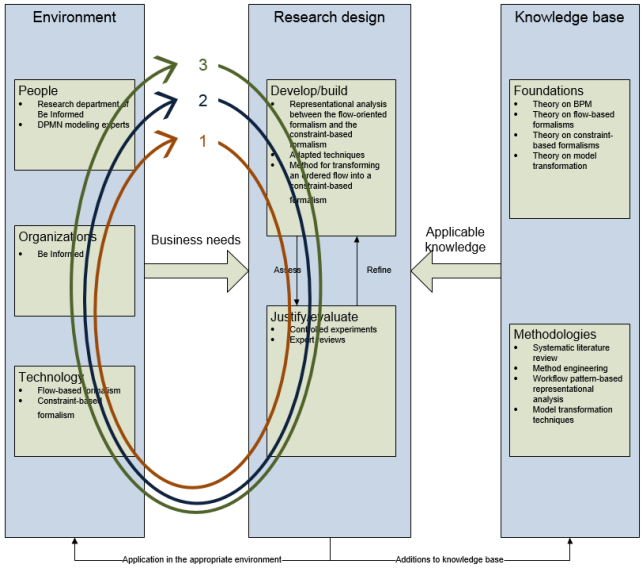

Figure 4: Hevner framework applied to this paper

Different methods of analysis conclude that BPMN offers the most functionality compared to other procedural formalisms (Recker et al., 2005; Wohed et al., 2006). Furthermore, it is identified that DPMN and EM-BRA²CE conform to all of the given declarative properties described in (Goedertier & Vanthienen, 2007). Due to practical reasons, DPMN was chosen as target formalism.

Three iterations through the framework are required to create the method for transforming BPMN models into the DPMN formalism. The goal of the first iteration is to adjust the techniques to make them suitable for handling BPMN and DPMN models. The techniques are adapted to allow BPMN as input and DPMN as output models while maintaining the intended function of each technique. Each technique is then evaluated using a controlled experiment. During the experiment it is determined how suitable each technique is for providing a syntactical transformation or interpreting the intention of the activity. Suitability is measured in coverage fit (Strong & Volkoff, 2010). More specifically, coverage fit for the graph transformation and Well-Formed BPMN techniques is the amount to which the transformed model conforms to the modeling requirements of the target formalism. In other words, coverage fit is measured by the percentage of syntactically correct transformations of the process models in the sample set. For the remaining techniques, coverage fit is measured by the percentage of correct and logical classifications. This is measured in precision and automatically calculated for the semantic process similarity, related cluster pair and Naïve Bayes classifier techniques. Precision for the lexical analysis of activity labels is determined manually.

After each technique is evaluated, the second iteration starts. This iteration is used to combine the adjusted techniques into a method. The method is evaluated in a second controlled experiment with different case study data. During this controlled experiment, coverage fit is measured by the percentage of syntactically correct transformations. The second iteration ends with optimizing the order and priority of techniques based on the results of the controlled experiment. Lastly, the third iteration is used to determine how suitable the method is for transforming the process models. Several BPMN process models from the second set of case study data are transformed using the method. The transformed process models are evaluated by DPMN modeling experts and are given a rating from 1-10. The experts have at least one year of modeling experience with DPMN and have been enrolled in multiple DPMN modeling courses. By averaging the ratings, enablement fit is determined (Strong & Volkoff, 2010). Lastly, if the expert review shows the transformation is not optimal, final improvements to the method can be made made.

103 BPMN process models were provided by a pension fund in the Netherlands to use as a sample for the first iteration. Furthermore, 10 BPMN process models from a health care institution were used for the second and third iteration. The models were modeled by both end users and modeling experts according to the BPMN 2.0 standard and contain all of the BPMN common and extended core concepts and several BPMN specialist set concepts (Recker, 2010). The pension fund process models consist of a total of 517 activities, which averages out to approximately 5 activities per process model. The health care institution models consist of a total of 43 activities.

The selection of techniques was based on the functionality they provide. A set of two techniques was selected for each type of functionality. Graph transformation (H. Ehrig et al., 2004) and Well-Formed BPMN (Dijkman et al., 2008) were selected because of their ability to consider the syntactical elements of a process model. The Related cluster pairs (Niemann et al., 2012) and Semantic process similarity (M. Ehrig et al., 2007) were selected because of their ability to calculate a similarity value and provide an interpretation on multiple concepts simultaneously. For interpreting the process model concepts, a Naïve Bayes classifier (Witten, Frank, & Hall, 2011) and lexical analysis of activity labels techniques are used. The Naïve Bayes classifier and lexical analysis of activity label techniques were selected to interpret the activity labels within a process model.

RESULTS

As has been mentioned previously, each technique needs to be adjusted for transforming BPMN process models to DPMN process models or to interpret the intention of the concepts. For the graph transformation technique, 59 out of the 103 BPMN process models from the first sample set were used as a learning set to determine which productions are required to perform the transformation. The remaining 44 process models were used to validate these productions. In total, 71% of the process models could be transformed. Pre-defined Well-Formed BPMN clusters were already available (Dijkman et al., 2008) and corresponding DPMN clusters were created. The corresponding clusters were validated for correctness by DPMN experts. In total, 32% of the process models could be transformed with the Well-Formed BPMN technique. However, 52 of the process models were discarded because they did not conform to the Well-formed BPMN rules stated in Dijkman et al. (2008). When accounting for this in the coverage fit calculation, the total amount of correctly transformed process models amounts to 64%.

The similarity measurement calculations for the related cluster pair and Semantic similarity techniques were programmed in MS Excel. For the related cluster pair technique an experiment was performed in which two set of queries were created to find similar clusters in the sample set of process models. The mean average precision for all queries combined is 0.55. Furthermore, 5 out of the 7 clusters manually identified as having a similar intention were indicated as similar. To determine the suitability for the semantic process similarity technique, a similar technique as in Dijkman, Dumas, Dongen, Reina and Mendling (2011) was used. Eight process models were randomly selected from the sample set as queries. Two process models were left unchanged. The labels of the second pair of process models were changed without changing the meaning of the labels. Half of the labels were removed from the query for the third pair of process models and lastly, the order of activities was changed for the last pair of process models. The mean average precision is for this technique is 0.64. Furthermore, four queries did not receive the highest similarity value for their corresponding process model.

To calculate the precision of the Naïve Bayes Classifier, activities which perform a "*create*" or "*send*" action on a document are classified as TRUE. When such an activity is found it is tagged. Depending on the rule for that tagged it can either be transformed, omitted or expanded. The sample data was preprocessed to make it suitable for classifying activities. First, every instance was given a Boolean variable, which is set to true if the activity creates or sends a document and is set to false otherwise. Secondly, all the activity labels were transformed into a vector. This resulted in 256 attributes on which the classification is based. The Naïve Bayes classifier is able to classify the activity labels as a "*create*" or "*send*" activity with a precision close to 99%.

For the lexical analysis of activity labels technique, the process labels in the sample set containing 103 process models were classified with the verb classification provided by Mendling et al. (2011). However, when trying to determine corresponding structures, the classification of the activity labels seemed ambiguous in several cases. Therefore, a new verb classification scheme was made based on the list provided by Mendling et al. (2011), the verbs occurring in the activity labels of the sample set and a list of most used verbs in DPMN. With the new classification scheme, 53% of labels are classified correctly. However, 100 of the activity labels do not contain a verb. Taking this into account, the coverage fit of this technique is 73%. The coverage fit for the techniques is summarized in Table 1.

Table 1: Coverage of techniques in percentages

TECHNIQUE	COVERAGE FIT
Graph Transformation	0.71
Well-Formed BPMN	0.64
Related Cluster Pair	0.55
Semantic Process Similarity	0.64
Naïve Bayes Classifier	0.986
Lexical Analysis of Activity labels	0.73

The results from the techniques as described in Table 1 do not provide any reason to leave out one of the techniques. However, due to time constraints it was not possible to create domain specific queries for the related cluster pair and semantic process similarity techniques for the next iterations. Therefore, no additional queries were created for these techniques.

To combine these techniques into a method, a distinction is made between techniques which perform a syntactic transformation and techniques that interpret the intention of the process model concepts. The transformation techniques were grouped and performed in a sequential manner. This already results in a process model in DPMN, but does not include any semantic interpretation. The techniques on the semantic level are performed simultaneously and are used to attach tags to the concepts in the transformed process model. Based on a priority of tags the tagged concepts were changed, expanded or omitted.

After the priority of tags was optimized, the method was used to transform 10 BPMN process models. 9 out of 10 process models could be transformed. One could not be transformed syntactically correct, because the method is not able to deal with loops. The looping structures can be transformed, but are not executable due to the way sequence flow between activities is transformed (Figure 5).

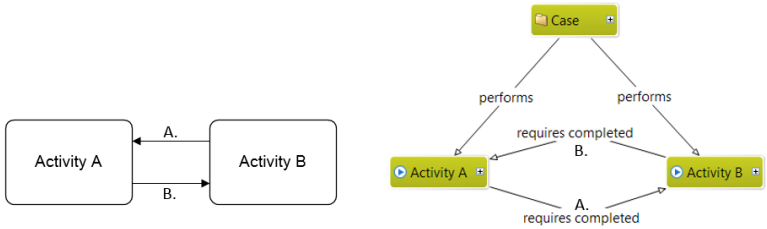

Figure 5: Example of a transformed loop structure.

The transformed process models were presented to three DPMN modeling experts. The experts were asked to rate each model for their ability to use the concepts of the target business process formalism effectively. The rated process models can be viewed in Table 2.

Table 2: Expert ratings of transformed process models

Model	Expert 1	Expert 2	Expert 3
#1	3	4	6
#2	3	3	6
#3	7	8	7
#4	4	5	5
#5	5	4	7
#6	7	4	7
#7	5	4	6
#8	9	5.5	8
#9	7	6	7

On average, the models were rated a 5.6 on a scale from 1 to 10. Cronbach's Alpha was used to assess consistency of the expert review. Generally, a score of 0.7 is considered sufficient when determining internal consistency. Cronbach's Alpha calculated for the results in Table 2 amounts to 0.77, which indicates the internal consistency of the expert review results is sufficient. The main feedback given by the experts is that there are many unnecessary constraints between activities, instead of constraints on artifacts resulting from activities. Furthermore, goal orientation is not properly implemented, since all activities have to be performed, instead of looking which activities need to be performed to reach the goal of the process model. Lastly, some activities are included in the target models which do not seem to serve any function. The method is currently not able to distinguish relevant and irrelevant activities, since none of the techniques are currently able to make this distinction. All experts agreed that the syntactical techniques deal with XOR waypoints very well. To deal with these deficiencies, an extra activity is added to the method. During the "Perform expert review" activity, unnecessary constraints between activities are removed or rearranged to artifacts and activities are made optional when possible. On average, 4.4 modeling actions are performed during the expert review. In comparison, creating the models from scratch requires 21.4 modeling actions. The resulting method can be viewed in Figure 6.

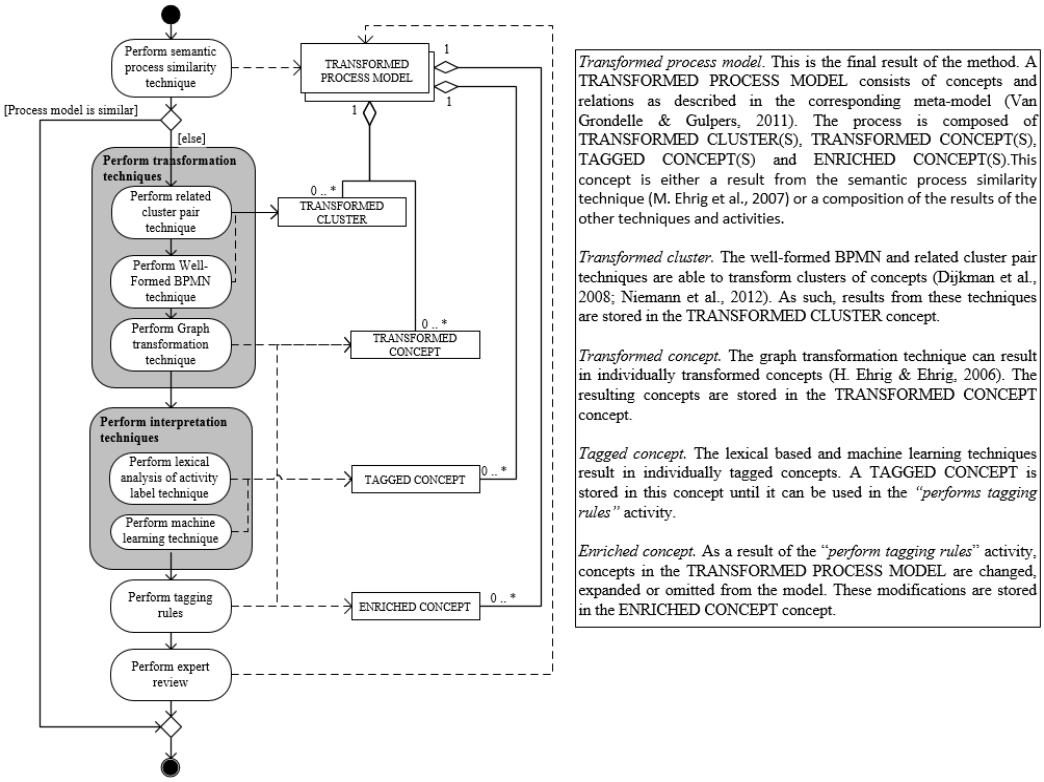

Figure 6: Resulting method and concept description

An example of a model transformation using this method can be viewed in Figure 7. The first step of the method is to check if a process model is semantically similar. Since no semantic similarity queries have been created for this example, the method continues to the "perform transformation techniques" phase. The related cluster pair technique is performed first. However, no additional queries clusters were made during the second and third iteration of development. As a consequence, no clusters are transformed using the related cluster pair technique. The next technique to be executed is the Well-formed BPMN technique. Concepts B, C, D and E conform to a predefined cluster and are transformed to the corresponding DPMN cluster. The remaining concepts (A, F and G) do not conform to any Well-Formed BPMN structure, thus the next technique is initiated. The remaining concepts are transformed using the Graph transformation technique. According to the productions developed in the first iteration, the concepts A, F and G are omitted from the target model. Since all concepts have been transformed during the transformation phase, the method continues with the next phase. During the next phase, interpretation techniques are performed. The machine learning technique classifies concept B as a "communicate" and "determine" activity concept. The lexical analysis technique tags concept B as a "determine" activity. Concept E is tagged as "determine" and "communicate" activity by the machine learning technique. The activity concept is also tagged as "communicate" activity by the lexical analysis technique. The last remaining activity concept, concept D, is tagged as "create document" and "determine" activity by the machine learning technique. Furthermore, it is tagged as "create document" by the lexical analysis technique. Now all activity process model concepts are tagged, the post processing phase is finished and the tagging rules are applied in the

next method activity. The concepts are enriched based on the tag with the highest priority. The enrichment of concept B consists of adding a decision concept. However, since the activity concept already contains a decision concept, no enrichment takes place for this concept. The corresponding structure of the tag with the highest priority of concept E does not contain any additional concepts. Therefore, concept E is left as is. Concept D is enriched by adding a "document" concept to the activity. Lastly, concept B is set to "perform if needed" during the expert review method activity to make the model goal-oriented.

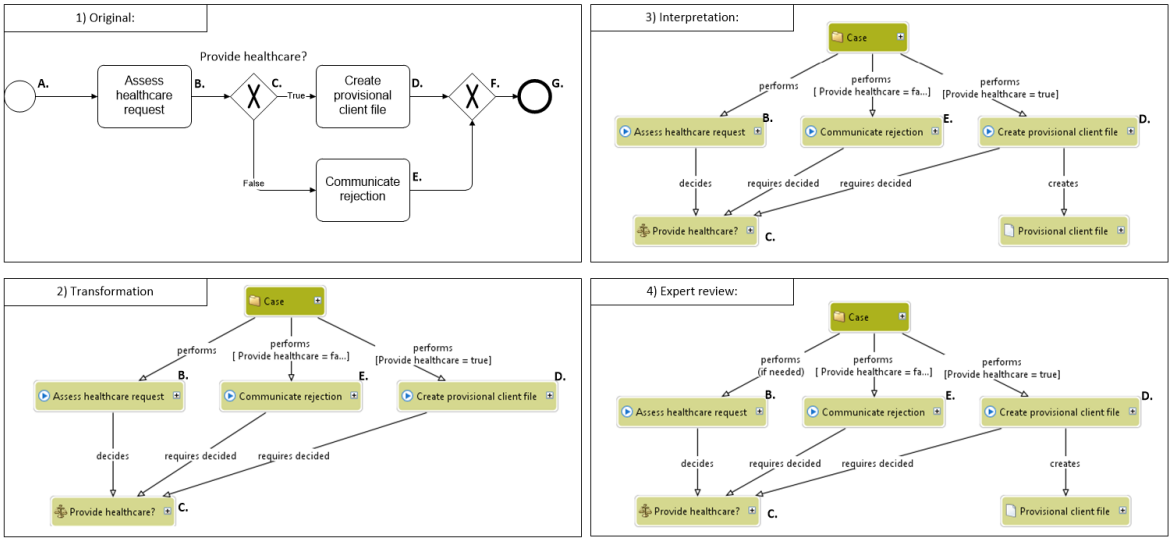

Figure 7: Example of a process model transformation

CONCLUSION

The contribution of this paper is twofold. First, this research shows a way to reduce the inevitable barrier between many organizations who adopted procedural formalisms and the adoption of a declarative approach to modeling processes. The results of this research can provide the grounds for further research on this subject. Second, a practical application for method engineering, model transformation and explication is presented in this paper.

The research presented in this paper has several challenges and limitations. First, it should be stated that the method created during this project is not regarded as a final product, but rather a step towards creating method which is updated as new techniques become available. As such, the techniques used in the method are not considered exhaustive. Currently, the method consists of six techniques, which can be increased in future research. For example, research by Dijkman et al. (2011), Awad, Polyvyanyy and Weske (2008), van der Aalst (2011), Yan & Han (2002) and Gao, Xiao, Tao, & Li (2009) can be used to expand the method. Lastly, research could be done on adding other formalisms as source or target formalisms. For example, UML activity diagrams could be added as source formalism and Condec as target formalism. This way, a framework can be created which is able to transform multiple source formalisms to multiple target formalisms with a custom selection of techniques.

Regarding the process of developing the method, several points can be made. First of all, the related cluster pair and semantic process similarity techniques were not tested in the method. Therefore, it is not guaranteed that these techniques are fully functional in a methodical transformation. In addition to this, the other techniques should be further experimented with as well. While the adjustment of the techniques was tested extensively, limited time and sample processes prevented us from further testing the techniques during the methodical transformation. When the method is further improved and techniques are added or removed, an application such as the Meta-environment (Brand et al., 2001) could be used to automatically perform this transformation.

The goal of this paper is to provide a methodical way of transforming BPMN models into DPMN models and to explicate the original business constraints of a process model. To realize this, a method was created which contains four transformation and two interpretation techniques. The techniques and method were adapted and validated using different sets of BPMN process models. The method is able to transform most of the BPMN models, except process models with loops. The resulting process models are considered sufficiently utilizing declarative advantages by DPMN modeling experts. Furthermore, an average of 4.4 manual modeling actions are required after the method is performed. In comparison, 21.4 modeling actions are required to create the models from scratch.

REFERENCES

Almeida, J., Eck, P., & Iacob, M. (2006). Requirements Traceability and Transformation Conformance in Model-Driven Development. In *Enterprise Distributed Object Computing Conference (EDOC'06). 10th IEEE International* (pp. 355–366). IEEE.

Awad, A., Polyvyanyy, A., & Weske, M. (2008). Semantic Querying of Business Process Models. In *Enterprise Distributed Object Computing Conference* (pp. 85–94). IEEE.

Brand, M. G. J. Van Den, Deursen, A. Van, Heering, J., Jong, H. A. De, Jonge, M. De, & Kuipers, T. (2001). The ASF + SDF Meta-Environment: a Component-Based Language Development Environment. In *Compiler Construction* (pp. 365–370). Springer Berlin Heidelberg.

Brinkkemper, S. (1996). Method engineering : engineering of information methods and tools. *Information and Software Technology*, *38*, 275–280.

Davenport, T. H., & Stoddard, D. B. (1994). Reengineering : Business Change of Mythic Proportions ? *MIS Quarterly*, *18*(2), 121–127.

Dijkman, R., Dumas, M., Dongen, B. Van, Reina, K., & Mendling, J. (2011). Similarity of Business Process Models : Metrics and Evaluation. *Information Systems*, *36*(2), 498–516.

Dijkman, R., Dumas, M., & Ouyang, C. (2008). Semantics and analysis of business process models in BPMN. *Information and Software Technology*, *50*, 1281–1294.

Ehrig, H., Prange, U., & Taentzer, G. (2004). Fundamental Theory for Typed Attributed Graph Transformation. In *Graph transformations* (pp. 161–177).

Ehrig, M., Koschmider, A., & Oberweis, A. (2007). Measuring Similarity between Semantic Business Process Models. In *Proceedings of the fourth Asia-Pacific conference on Comceptual modelling* (pp. 71–80).

Gao, X., Xiao, B., Tao, D., & Li, X. (2009). A survey of graph edit distance. *Pattern Analysis and Applications*, *13*(1), 113–129.

Goedertier, S., Haesen, R., & Vanthienen, J. (2007). *EM-BrA2CE v0. 1: A vocabulary and execution model for declarative business process modeling* (pp. 0–74). Retrieved from http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1086027

Goedertier, S., & Vanthienen, J. (2007). Declarative Process Modeling with Business Vocabulary and Business Rules. In *On the move to meaningful internet systems* (pp. 603–612).

Hammer, M. (1990). Reengineering Work: Don't Automate, Obliterate. *Harvard Business Review*, *68*(4), 104–112.

Hevner, A. R., Ram, S., March, S. T., & Park, J. (2004). Design Science in Information Systems Research. *MIS quarterly*, *28*(1), 75–105.

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems*, *15*, 251–266.

Mendling, J., Recker, J., & Reijers, H. (2011). On the usage of labels and icons in business process modeling. *Expert Systems With Applications*, *38*(6), 7029–7049.

Niemann, M., Siebenhaar, M., Schulte, S., & Steinmetz, R. (2012). Comparison and Retrieval of Process Models using Related Cluster Pairs. *Computers in Industry*, *63*(2), 168–180.

OMG. (2011). *Business Process Model and Notation (BPMN). Version 2.0*. Retrieved from http://www.omg.org/spec/BPMN/2.0/

Peffers, K., Tuunanen, T., Rothenberger, M. a., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77.

Pesic, M., & van der Aalst, W. M. P. (2006). A Declarative Approach for Flexible Business Processes Management. In *Business Process Management Workshops* (pp. 169–180). Springer Berlin Heidelberg.

Peterson, J. L. (1981). *Petri Net Theory and the Modeling of Systems*. *Englewood Cliffs NJ PrenticeHall Inc* (Vol. 24, p. 290). Prentice-Hall.

Recker, J. (2010). Opportunities and Constraints : The Current Struggle with BPMN. *Business Process Management Journal*, *16*(1), 181–201.

Recker, J., Indulska, M., Rosemann, M., & Green, P. (2005). Do process modelling techniques get better? A comparative ontological analysis of BPMN. In *16th Australasian Conference on Information Systems*.

Strong, D. M., & Volkoff, O. (2010). Understanding organization-enterprise system fit: a path to theorizing the information technology artifact. *MIS quarterly*, *34*(4), 731–756.

Tan, P.-N., Steinbach, M., & Kumar, V. (2006). *Introduction to data mining*. Pearson Addison-Wesley.

Van de Weerd, I., & Brinkkemper, S. (2008). Meta-modeling for situational analysis and design methods. In M. R. Syed & S. N. Syed (Eds.), *Handbook of research on modern systems analysis and design technologies and applications* (pp. 38–58). Hershley, Idea Group Publishing.

Van der Aalst, W. M. P. (2011). *Process Mining*. Springerverlag Berlin Heidelberg.

Van der Aalst, W. M. P., & ter Hofstede, A. H. M. (2005). YAWL: yet another workflow language. *Information Systems*, *30*(4), 245–275.

Van der Aalst, W. M. P., Weske, M., & Grünbauer, D. (2005). Case handling: a new paradigm for business process support. *Data & Knowledge Engineering*, *53*(2), 129–162.

Van Grondelle, J., & Gulpers, M. (2011). Specifying Flexible Business Processes using Pre and Post Conditions. In *Practice of Enterprise Modeling* (Vol. 92, pp. 1–14).

Van Grondelle, J., Zoet, M., & Vermeer, F. (2013). Characterizing the Declarativity of Business Process Formalisms. In *24th International Information Management Association Conference*. New York.

Verschuren, P. J. M., & Doorewaard, H. (2007). *Het ontwerpen van een onderzoek*. Lemma.

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques* (p. 376). Morgan Kaufmann.

Wohed, P., van der Aalst, W. M. P., Dumas, M., ter Hofstede, A. H. M., & Russel, N. (2006). On the suitability of BPMN for business process modelling. In *4th International Conference on Business Process Management* (pp. 161–176). Springer Berlin Heidelberg.

Yan, X., & Han, J. (2002). gSpan: Graph-Based Substructure Pattern Mining. In *IEEE International conference on Data Mining* (pp. 721–724). IEEE.

Zur Muehlen, M., & Recker, J. (2008). How Much Language is Enough ? Theoretical and Practical Use of the Business Process Modeling Notation. In *20th International Conference on Advanced Information Systems Engineering* (pp. 465–479). Springer Berlin/Heidelberg.

COPYRIGHT