

Capturing Ordered Flow in a Constraint-Based Formalism

Floor Vermeer

Thesis

31 July 2013

Bicycling directions to Neude



Princetonlaan

1. Head west on Princetonlaan toward Sorbonnelaan



2. Turn left onto Sorbonnelaan



3. Turn left to stay on Sorbonnelaan



4. Turn right toward Biltstraat

5. Take the pedestrian tunnel



6. Turn left toward Biltstraat



7. Turn right onto Biltstraat

8. Continue onto Wittevrouwenbrug

9. Continue onto Wittevrouwenstraat

10. Continue onto Voorstraat



11. Keep right to stay on Voorstraat

12. Continue onto Neude



Neude



Title : **Capturing Ordered Flow in a Constraint-Based Formalism**

Study : Business Informatics

Document : Master Thesis

Version : 1.0

Date : 31-07-2013

Author: : Floor Vermeer
: Floorvermeer@gmail.com
: University Utrecht

Daily Supervisor: : Martijn Zoet, MSc
: martijn.zoet@hu.nl
: University of Applied Sciences Utrecht

1st Supervisor: : dr. Slinger Jansen
: slinger.jansen@uu.nl
: University Utrecht

Company supervisor: : Jeroen van Grondelle, MSc
: j.vangrondelle@beinformed.com
: Be Informed, Apeldoorn



Universiteit Utrecht



be informed

Preface

Two years ago, after receiving my bachelor's degree from the University of Applied Sciences in Utrecht, I decided to continue my academic career and enroll for the master's program Business Informatics at Utrecht University. During the course of the master's program I have gained knowledge and skills as an academic researcher. As such, this document is the final test of aptitude as an academic researcher.

This document contains the results of my graduation research project for my master degree in business informatics at Utrecht University. The research project was performed at Be Informed in cooperation with Utrecht University. During the six months it took to perform the research, a lot of interest was shown and feedback was received on the differences between procedural and declarative process models and how to transform them. I hope this research contributes to the knowledge on the subject and interests people in exploring the possibilities for model transformation.

Acknowledgements

First of all, I would like to thank Martijn Zoet for his valuable advice on matters related and unrelated to this research project. His advice helped me finish this research in its current state and provided insight in career possibilities for when I finish this study. Secondly, I would like to thank Jeroen van Grondelle for his inspirational and helpful ideas during the many brainstorm sessions we had. My thanks also go out to Jouri Fledderman for helping with some of the technical difficulties during this research.

Furthermore, my appreciation goes out to Eline, Thomas, Frank, Penny and Bart for sharing the successes and disappointments during our time at Be Informed. The conversations were often inspiring, motivating and a welcome distraction.

Lastly, I would like to thank my family, friends and everyone who supported me during this research. Without them, I wouldn't be able to get this far.

A moment of success during this research was a co-authorship on a paper accepted for IIMA 2013. With this paper, we aim to increase the understanding of declarativity in process models. Furthermore, a second paper with the results of this research will be submitted to the ACIS 2013 conference.

Abstract

Many organizations use business process management to manage and model their processes. For the purpose of modeling business processes, two different types of modeling formalisms can be distinguished: Flow-based formalisms, resulting in procedural process models and constraint-based formalisms, resulting in declarative process models.

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. In contrast, declarative process models are recognized by explicit business constraints and the execution scenario is inferred from these business constraints. Several limitations of procedural models have been discovered in recent literature, which can be dealt with using a constraint-based formalism. Flow-based formalisms such as BPMN are considered the current standard for business process modeling. Consequently, a large amount of time and effort has already been spent by organizations on modeling their processes with a flow-based formalism. As such, the goal of this research is to develop a methodical way of transforming procedural process models into a constraint-based formalism. Furthermore, to alleviate the limitations induced by procedural formalisms, the process model concepts are interpreted in order to explicate the original business constraints it was based on.

To realize this, six techniques were selected: Graph transformation, Well-formed BPMN transformation, related cluster pair similarity measurement, semantic process model similarity measurement, lexical analysis of activity labels and a Naïve Bayes classifier. These techniques were adjusted to be compatible with BPMN as source formalism and the Declarative Process Modeling Notation (DPMN) as target formalism. The techniques were tested using a sample set containing 103 BPMN process models. Resulting from the techniques was a set of transformation and interpretation rules, which were used to create a transformation method. The method was used to transform and interpret a different sample set, which contains 10 BPMN process models. An evaluation on the result of these transformations was used to improve and fine-tune the method.

The method is able to transform 9 out of 10 process models. The 9 successfully transformed process models were reviewed by DPMN modeling experts and received an average grade of 5.6 (on a scale from 1 to 10) for utilizing the declarative properties in an advantageous manner. Based on the results of this review, final improvements were made to the method.

In future research, additions can be made to the method. The method can be expanded with multiple source and target formalisms. Additional techniques can be added as well. For example, graph mining, graph edit distance, process mining, process behavior and BPMN-Q are considered as possible additions to the set of techniques.

Contents

1	Introduction.....	1
1.1	Research triggers and problem definition.....	2
1.1.1	Scientific Triggers.....	2
1.1.2	Practical Triggers	3
1.1.3	Business Triggers	3
1.1.4	Problem statement.....	4
1.2	Research Question	4
1.3	Research Method	5
1.4	Research Model.....	7
1.4.1	The research model.....	8
2	Literature review	10
2.1	Business Process Management	10
2.2	Flow-based formalisms.....	11
2.3	Constraint-based formalisms	13
2.4	Declarative properties mapped to process modeling formalisms	15
2.5	Quality characteristics in different types of formalisms	16
2.6	Transformation techniques	17
2.6.1	Graph transformation.....	17
2.6.2	Transforming Well-formed BPMN.....	19
2.6.3	Related cluster pair technique	20
2.6.4	Measuring semantic similarity between process models	23
2.7	Explication techniques.....	25
2.7.1	Lexical analysis of activity labels	25
2.7.2	Classifying process fragments using machine learning	25
3	Data collection.....	28
3.1	Unit of analysis 1: Techniques.....	29
3.2	Unit of analysis 2: Method	29
4	Data analysis.....	30
4.1	Techniques	30
4.1.1	Graph transformation.....	30
4.1.2	Well-formed BPMN transformation.....	31
4.1.3	Related cluster pair transformation	32
4.1.4	Measuring semantic similarity between process models	36
4.1.5	Classifying process labels using machine learning	37

4.1.6	Lexical analysis of activity labels	38
4.1.7	Evaluation of techniques	39
4.2	Method description	39
4.2.1	Product Deliverable Diagram	40
4.2.2	Activity table	40
4.2.3	Concept table	41
4.2.4	Transformation results	42
4.3	Final method	43
4.3.1	Final Product Deliverable Diagram	45
4.3.2	Added activities	45
5	Discussion	46
6	Conclusion	47
7	References	49
	Appendix A: Systematic literature review protocol	55
	Appendix B: Workflow patterns	57
	Appendix C: Code used for calculating similarity values	65
	Appendix D: Well-formed transformation technique rules	73
	Appendix E: Graph productions	80
	Appendix F: Verb classifications	83
	Appendix G: Post-processing rules	84

List of figures

Figure 1: Transformation and interpretation of a process model. Adapted from (Pesic & van der Aalst, 2006).....	5
Figure 2: Conceptual framework applied to this research.....	6
Figure 3: Design research method. Retrieved from Peffers et al. (2007).	7
Figure 4: The research model.....	8
Figure 5: Coverage and enablement fit.....	9
Figure 6: Example of a BPMN process model.....	12
Figure 7: Example of the Declarative Process Modeling Notation. Adapted from Van Grondelle & Gulpers (2011).	14
Figure 8: Mapping between concepts of a procedural and declarative formalism.	14
Figure 9: Example of a graph transformation.	18
Figure 10: Example of production "REFTASK".	18
Figure 11: Example of production "REFSEQUENCE".	19
Figure 12: Well-formed BPMN clusters.....	19
Figure 13: Instances in a feature space.	26
Figure 14: Several iterations through the research framework.....	28
Figure 15: Example of a transformation.....	31
Figure 16: Transformation of cluster 1.....	33
Figure 17: Example of a related cluster pair transformation.	35
Figure 18: Another example of a related cluster pair transformation.....	35
Figure 19: Method for transforming procedural models.	40
Figure 20: Example of transformation steps and resulting process model.	42
Figure 21: Example of the expert review step.	44
Figure 22: Method for transforming procedural models.	45
Figure 23: A framework for business process model transformations.	46

List of tables

Table 1: Properties of flow-based and constraint-based formalisms. Adapted from Goedertier and Vanthienen (2007).....	11
Table 2: An overview of support for workflow patterns by several flow-based formalisms. Adapted from Wohed et al. (2006).....	12
Table 3: Declarative properties proposed by Goedertier & Vanthienen (2007).....	15
Table 4: Declarative Characteristics of Several Process Modeling Formalisms.	16
Table 5: Quality characteristics in different types of process modeling formalisms.	17
Table 6: Example cluster transformations.....	20
Table 7: similarity measurements and weights for extended Petri Net models.....	24
Table 8: Example of a C4.5 dataset.	26
Table 9: Example of a Naïve Bayes classifier sample set.....	27
Table 10: Example productions for the graph transformation technique.	30
Table 11: Reasons for not being able to transform a model.....	30
Table 12: Transformation rules for the well-formed transformation technique.....	31
Table 13: Errors in the Well-formed transformation technique.	32
Table 14: Query 1 for the related cluster pair technique.....	33
Table 15: Query 2 for the related cluster pair technique.....	34
Table 16: Results of the cluster retrieval experiment.	34
Table 17: Query 1 for the related cluster pair technique.....	35
Table 18: Query 2 for the related cluster pair technique.....	36
Table 19: Query 3 for the related cluster pair technique.....	36
Table 20: Query 4 for the related cluster pair technique.....	36
Table 21: Context elements considered for this experiment.....	37
Table 22: Similarity values for the randomly selected process models.	37
Table 23: Correctness percentages for the most common classifier algorithms.....	38
Table 24: Verb classification scheme.	39
Table 25: Activity table for the PDD.	41
Table 26: Concept table for the PDD.....	41
Table 27: Activity table for the final PDD.	45

1 Introduction

Currently, many organizations use the Business Process Modeling Notation (BPMN) or similar flow-based formalisms to model their processes (Recker, 2008, 2010; Zur Muehlen & Recker, 2008). BPMN is a standard of the Object Management Group's (OMG) for process modeling (White, 2004). It was introduced in 2004 and the current version (2.0) was released in 2011 (OMG, 2011). BPMN provides a graphical notation for specifying business processes based on a flowcharting technique similar to activity diagrams from Unified Modeling Language (UML).

While the usage of flow-based process formalisms continues, several problems with these types of formalisms have been identified (Pesic & van der Aalst, 2006; van der Aalst & Jablonski, 2000; van der Aalst, Weske, & Grünbauer, 2005; Van Grondelle & Gulpers, 2011). Van der Aalst et al. (2005) identify four problems with flow-based formalisms: First of all, 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. Lastly, traditional flow 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 traditional formalisms are inefficient in dealing with a rapidly changing and complex business environment. In traditional workflow management systems, making changes can be time consuming and complex. Furthermore, dealing with a dynamic environment requires a large amount of process variants, which overcomplicates the process models.

Multiple attempts have been made to solve one or more of these issues (Kammer, Bolcer, Taylor, Hitomi, & Bergman, 2000; Kim, Choi, & Park, 2011; Lu, Sadiq, & Governatori, 2009; Reichert & Dadam, 1998; Smirnov, Reijers, Weske, & Nugteren, 2012; Weber, Reichert, Mendling, & Reijers, 2011; Weske, 2001). For example, Kim et al. (2011) propose a method for dealing with exceptions in business processes. In their method, an automated system predicts exceptions and gives suggestions on how to deal with the exception. The suggestion is then approved by an exception manager. In Weber et al. (2011) refactoring is applied to process models. Refactoring originates from the domain of software engineering and refers to the restructuring of software code without changing the functionalities of the software (Fowler & Beck, 1999). Kammer et al. (2000) formulate eleven functionalities for workflow systems that are required to deal with exceptions and a dynamic environment. Lastly, Smirnov et al. (2012) propose a framework that can be used to simplify process models by abstracting certain parts of the model.

We consider the solutions mentioned above as workarounds for problems that are inherent of flow-based formalisms. Other research proposes a declarative approach to solving these issues (Lu et al., 2009). In their framework, Lu et al. (2009) propose a business process constraint network which is used to contain process tasks and constraints. The constraints are used to determine the process per instance. This prevents the organization from prescribing processes at design time and instead looks at the best fitting process variant at execution time. However, this solution is still focused on selecting the best ordered flow and therefore does not deal with the issues mentioned by Van der

Aalst et al. (2005), which are described earlier in this section. A solution to these issues might require a paradigm shift towards constraint-based formalisms (van der Aalst et al., 2005).

As has been described earlier, many organizations apply flow-based formalisms like BPMN to manage their processes. These types of formalisms have several limitations, which were previously mentioned in this chapter. Furthermore, this chapter also indicates some research trends towards a formalism that deals with these limitations. However, organizations already have a large amount of time and resources invested in flow-based models and designs. Therefore, a research question is formulated for determining how to transform ordered flows into a constraint-based formalism. The research question can be found in section 1.2. Chapter 2 contains the current literature on the subject of this thesis. Chapter 3 elaborates on what and how data is collected. In chapter 4 the results of the research are presented. Firstly, the adjusted techniques and their performance are explained. Then, the initial method is described and evaluated. Lastly, the adjustments needed for the final method are described and the final method is depicted. The results of the thesis are followed by a discussion and conclusion, chapter 5 and 6 respectively.

1.1 Research triggers and problem definition

This section gives a description of the scientific, practical and business triggers of this research. The triggers are combined into a problem statement at the end of this section.

1.1.1 Scientific Triggers

Many research in the field of Business Process Management (BPM) has focused on flow-based formalisms (Recker, Indulska, Rosemann, & Green, 2005; van der Aalst & ter Hofstede, 2005; Weske, 2007; Wohed, van der Aalst, Dumas, ter Hofstede, & Russel, 2006). This research has resulted in multiple flow-based formalisms, such as BPMN, Petri Net, Colored Petri Net, Event Driven Process Chains, Interaction Flow diagrams, Workflow Nets and Graph based Workflow Language (Weske, 2007). With the abundance of flow-based formalisms, some researchers resort to giving their procedural formalisms names like “Yet Another Workflow Language” (YAWL) (van der Aalst & ter Hofstede, 2005). These flow-based formalisms have several limitations when it comes to dealing with complex processes and changing business environments (van der Aalst & Jablonski, 2000; Van Grondelle & Gulpers, 2011). Flow in these formalisms requires a separate fork for every exception or alternative to the standard path, resulting in redundancy and overly complex models. An alternative is creating a separate flow model for every variant or exception, which also results in redundancy and complexity. Furthermore, these types of flow cause actual and complex work packages to be restricted into activities, combine distribution and authorization of activities (which should not coincide), the context is not taken into account and it focuses on what should be done instead of what can be done (van der Aalst et al., 2005).

Recent research has made a shift towards a more constraint-based paradigm (Kardasis & Loucopoulos, 2004; van der Aalst et al., 2005). The Manchester Business Rules Management formalism uses a goal oriented framework for managing business rules (Kardasis & Loucopoulos, 2004). Van der Aalst et al. (2005) propose a constraint-based formalism by focusing on case handling. Case handling solves the limitations of flow-based formalisms identified in (van der Aalst et al., 2005). The constraint-based formalisms by van Grondelle & Gulpers (2011) and Goedertier, Haesen, & Vanthienen (2007) offer flexibility by using pre- and post-conditions to determine among others when, how and by whom an activity should be performed and what the results of the activity are.

Summarizing, in the BPM and BRM fields more attention is going towards constraint based formalisms. While there are already several models available, more research is needed on these types of formalisms. This opportunity is used for this thesis by providing research on this topic.

1.1.2 Practical Triggers

As has been determined in the previous paragraphs, flow-based formalisms have several limitations in capturing complex and dynamic processes. As a practical consequence, a large amount of splits and variants are often required to model all context-specific parts. An alternative to creating splits and variants is making a separate model for each specific context, but this causes similar manageability problems.

Furthermore, most order in flow-based formalisms is modeled within the business constraints and formalism requirements. However, the business constraints itself are not modeled and this causes the actual requirements and business constraints to be moved to the background. When changes to the process model are required, traceability to the original requirements might have become lost (Almeida, Eck, & Iacob, 2006). If changes are made to the model, it is unclear if they violate the original requirements the model was built within.

As can be concluded from the triggers described in the previous paragraphs, flow-based formalisms have several limitations in capturing highly complex business environments. These limitations are reduced when using a constraint-based formalism to construct flows (van der Aalst et al., 2005). Therefore, the use and adoption of constraint-based formalisms seems pivotal. However, to the knowledge of the authors there is currently no method for transforming the widely adopted flow-based formalisms to constraint based formalisms. One of the goals of this research is to provide a practical solution for this obstacle.

1.1.3 Business Triggers

Organizations are currently developing support for modeling business processes, which is able to deal well with the inherent misfit between ordered flows and constraint-based formalisms (Van Grondelle & Gulpers, 2011). The constraint-based formalisms fit the requirements of their customers: Flexible processes that can be tailored to an individual customer, customizability of the ordered flow within the scope of the constraints by experts and the traceability of design decisions in the ordered flows.

Furthermore, Jim Sinur, a market expert from Gartner says about business rules: *"Business Rules and the technologies that enable them will help businesses cope with this new season of continuous business change"* (Sinur, 2012). Moreover, a recent whitepaper from Jeroen van Grondelle, research director at Be Informed, states: *"Conventional flow-oriented business process management approaches were developed for the old model and struggle with these changing circumstances"* (Van Grondelle & Rensen, 2013).

While the market is steadily moving towards a preference for business rules and constraint-based formalisms, a large amount of resources have already been invested in procedural models and designs by many organizations (Recker, 2008, 2010; Zur Muehlen & Recker, 2008). For them it is important to not let this investment go to waste. Therefore, it is necessary to provide these organizations with a clear and methodical approach for making the transition to a constraint-based formalism.

1.1.4 Problem statement

The triggers described above show an increase in attention and recognition for constraint-based formalisms as an alternative for modeling business processes. Furthermore, a practical and business need for adopting constraint-based formalisms can also be identified. Currently, most businesses have adopted a flow-based formalism to model their business processes. The authors of this thesis are not aware of a method aimed at transforming the flow-based formalisms to constraint based formalisms. This poses a problem that this thesis will attempt to solve. Taking the previous statements into account, the following problem statement is formulated:

“To preserve economic investments in flow-based business process models a methodical transformation into a constraint-based formalism is required”.

In the next section, a formal research question is formulated based on the problem statement.

1.2 Research Question

Continuing with the problem statement in the previous section, a method that is able to transform flow-based process models into a constraint-based formalism needs to be created. Before a research question can be formulated, it needs to be acknowledged that constraint-based formalisms are based on the business constraints of the organization for which the process is modeled. Furthermore, it needs to be acknowledged that existing ordered flow models are designed within the business constraints of the concerning organization. Based on these acknowledgements the following research question is formulated:

“How can formal underlying business constraints be extracted from procedural business process models?”

To answer the research question, a method that supports the transformations between formalisms is required. While tools and techniques have been created and used to transform data from one formalism to another (Decker, Dijkman, Dumas, & García-Bañuelos, 2008; H. Ehrig & Ehrig, 2006; Giner, Torres, & Pelechano, 2007; Niemann, Siebenhaar, Schulte, & Steinmetz, 2012), the authors of this thesis are not aware of the existence of an explicit method to perform such a transformation. Therefore, A method will be created using the method engineering approach (Brinkkemper, 1996; van de Weerd & Brinkkemper, 2008). To determine the contents of the method, several sub-questions are formulated.

First, the differences and similarities between the formalisms need to be identified. These can be identified by performing a representational analysis on the forms of representation and underlying concepts. As such, the first sub-question is: *“What are the representational differences between flow-based formalisms and constraint-based formalisms and how are their concepts related?”*

After answering the first sub-question the differences between the types of formalisms are known and a transformation between these formalisms is required. To transform all aspects of the source model to the target model, multiple transformation techniques are selected that cover different aspects of the source model. As such, the second sub-question concerns the transformation from a flow-based process model to a constraint-based process model. The second sub-question can be formulated as follows: *“How can a flow-based process model be transformed into a constraint-based process model?”*

However, a transformation alone does not suffice to alleviate the limitations of a procedural model (Figure 1, A and B), since the procedural limitations are transformed along with the rest of the model. A procedural model is 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 solution might be to interpret the process model concepts in order to explicate the original business constraints (Figure 1, C). As such, the third sub-question is formulated as follows: “How can business constraints be explicated from a transformed process model?” A second selection of techniques will be used for this purpose.

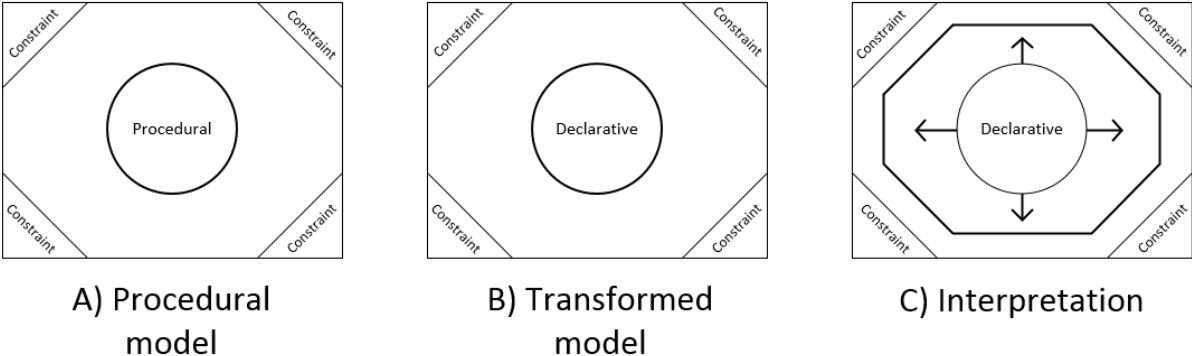


Figure 1: Transformation and interpretation of a process model. Adapted from (Pesic & van der Aalst, 2006).

1.3 Research Method

The artifacts resulting from the research are techniques and a method. Since artifacts 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 that can be used to construct a design research project in information system research. The conceptual framework applied to this research can be viewed in Figure 2. It contains a knowledge base, research design and an environment. The knowledge base provides the foundation on which the research is build, such as related literature and business process formalisms. Appropriate application of the knowledge base to the research design creates rigor in the research. The research design consists of two phases, namely the *develop & build* and the *justify & evaluate* phase. Obviously, the *develop & build* phase is used to create artifacts, whilst the *justify & evaluate* phase is used to assess created artifacts. Lastly, the environment is used to determine the scope in which the problem is solved.

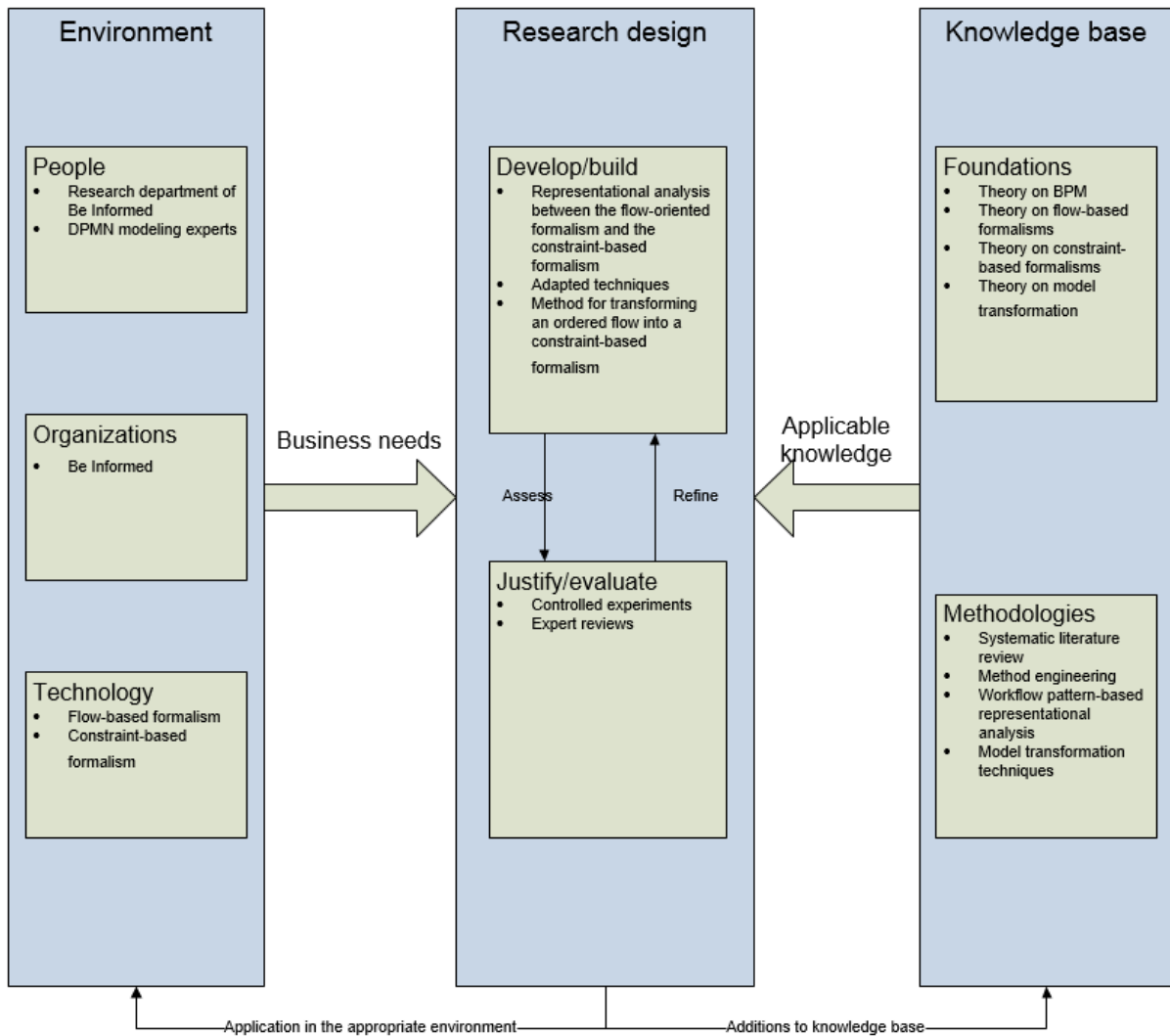


Figure 2: Conceptual framework applied to this research.

The seven guidelines for design research, described in Hevner et. al. (2004) are as follows:

1. Design an artifact. The result of an information system design research should be an artifact that solves a certain organizational problem. In this research, several transformation techniques are adapted and a method is created. The artifacts can be used by organizations to transform ordered flow to a constraint-based formalism.
2. Problem relevance. The research should be relevant to a constituent community. To make sure the research is relevant to the community. To do that, it should take the problems of the community into account and look at the people, organizations and IT of that community (Environment segment in Figure 2). The relevance of this research is explained in paragraph 1.1 of this document.
3. Design evaluation. The artifacts that are created throughout the research should be extensively evaluated and tested using the appropriate methods. The artifacts created during this research will be evaluated using controlled experiments and an expert review. The experiments also serve as a demonstration of the techniques and final method. Based on the results of the controlled experiments, refinement of the artifacts will be performed.
4. Research contributions. The research should contribute knowledge to the related scientific field. The contribution of this research is explained in chapter 6 (conclusions) of this document.

5. Research rigor. The research should include rigorous methods. This research employs methods for creating the artifacts that have already been used in published scientific work. For a full overview of methods used during this thesis, please see the sections below.
6. Design as a search process. The research should be performed on an iterative basis. This enables the researched to search for the best solution to the research problem. This research is divided in several phases and several evaluation loops are included, which make iteration possible. Furthermore, the principles of scrum are applied during the execution of this research, which is inherently iterative.
7. Communication of research. The research should be communicated to researchers and practitioners. A scientific article will be written based on the results of this research. For practitioners, a booklet will be made that contains the results of this research.

1.4 Research Model

In the previous section the constructs required to perform this research are mentioned. In this section, these constructs are used to create a research model. In order to create a rigid research model, literature is used from Peffers, Tuunanen, Rothenberger and Chatterjee (2007), Verschuren and Doorewaard (2007) and the guidelines provided by Hevner et al. (2004). Peffers et al. (2007) based their method on seven influential papers in the field of design research. According to Peffers et al. (2007), design research consists of six phases, namely the problem identification & motivation, defining the objective of the solution, design & development, demonstration, evaluation and communication phases. The problem identification and motivation phase is used to define the research problem and motivate why the problem should be solved. During the defining of the objectives of the solution phase, the research problem is translated in which objectives the expected solution should accomplish. The design and development phase encompasses the actual creation of the artifacts. Artifacts resulting from a design research can be constructs, methods, frameworks or instantiations (March & Smith, 1995). The demonstration phase is used to show that the artifact created functions by applying it to one or more problems. Methods suggested for demonstrations are case studies, simulation, experimentation or proof. During the evaluation phase the results of the demonstration are compared with the initial objectives and it is determined if the artifact performs as it was meant to. The last phase is the communication phase. During this phase, the importance and necessity of the artifact is communicated to researchers and practitioners. The phases of the design research method are depicted in Figure 3.

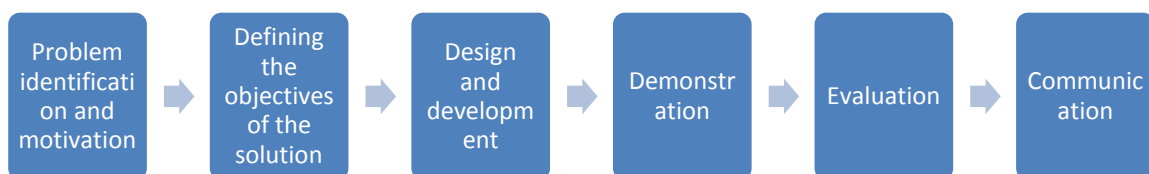


Figure 3: Design research method. Retrieved from Peffers et al. (2007).

Peffers et al. (2007) acknowledge that the nature of a design research can differ. Therefore, they explain different starting points of a design research. If a problem is observed or a design research is suggested as a result of earlier research, the design research project starts at the first phase. A design research can start in the second phase if there is an industry or research need for an artifact. When an artifact already exists, but isn't yet formally researched and linked to a problem, the design

research can be started in phase 3. Finally, a design research can start in phase 4 if a practical solution to a problem is observed. Research will focus on applying rigor to the solution in retrospect. This research starts in phase one of the design research method, since a problem is identified (See problem statement, section 1.1.4).

1.4.1 The research model

Based on the design research method by Peffers et al. (2007) and the theory on developing a research model by Verschuren and Doorewaard (2007) the research model for this thesis is created (Figure 4). The phases in the research model are explained below the figure.

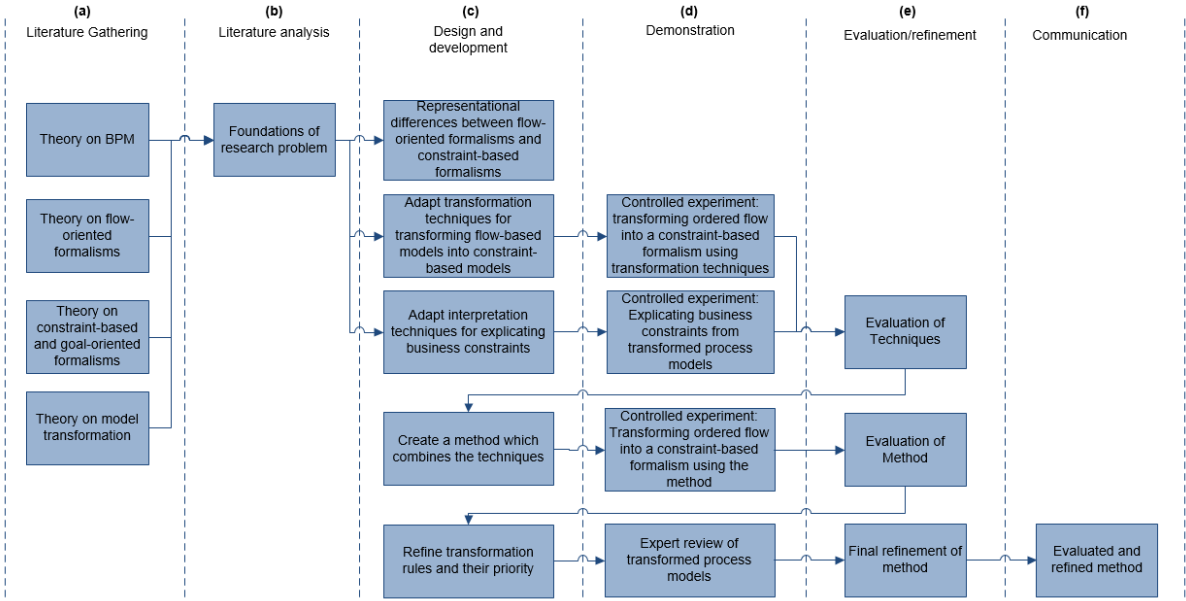


Figure 4: The research model.

The foundation of this research is based on theory relating to business process management, flow-based formalisms, constraint-based formalisms and existing model transformation techniques. In order to retrieve relevant literature on the subject of this thesis, a systematic literature review is performed at the beginning (phase a). The systematic literature review is based on the steps described by Okoli & Schabram (2010) and the description provided by Webster and Watson (2002). These methods were chosen because they are specifically aimed at information systems research, which is closely related to the subject of this thesis. A literature review protocol was created to guide the systematic literature review. The protocol can be viewed in Appendix A.

In regard to answering the first sub-question, the systematic literature review is expected to provide enough information for answering this question. In order to answer this question, past literature about differences between the types of formalisms is used. An example of the differences is also given by modeling control flow patterns in both types of formalisms. The workflow patterns can be found in Appendix B.

For the second and third sub-question a selection of techniques is made. These techniques were found based on a broad literature search and have already been applied in model transformation and interpretation or are expected to be useful for this purpose. Controlled experiments will be performed with each of these techniques to determine which are suitable for the transformation

between the two formalisms (phase d) or for the explication of business constraints. During the controlled experiments, the techniques are applied on a sample set of BPMN process models. The results of the controlled experiments are (partially) transformed process models or tags on the transformed process model concepts. The tags are applied based on an interpretation of the intention of these concepts. The techniques are evaluated on their coverage fit (Figure 5)(Strong & Volkoff, 2010). Coverage fit is defined as the amount of transformed process models that conform to the modeling requirements of the target formalism. A more detailed description on how the techniques are evaluated can be found in chapter 3.

After the evaluation of the techniques, a second iteration of design is initiated by creating the method. The method includes the techniques considered suitable for model transformation or constraint explication. This method will be subject to another controlled experiment with a sample set containing BPMN process models from a different source than was used in the experiments for the techniques. Based on the evaluation of the resulting transformations, the transformation rules and their priority are refined. Lastly, the final method will be used to transform several flow-based models. The transformed models are presented to experts in the target formalism to determine the enablement fit (Strong & Volkoff, 2010). Enablement fit is to what amount the transformation uses the concepts of the target business process formalism effectively. Based on the results of the expert review final improvements to the method are made.

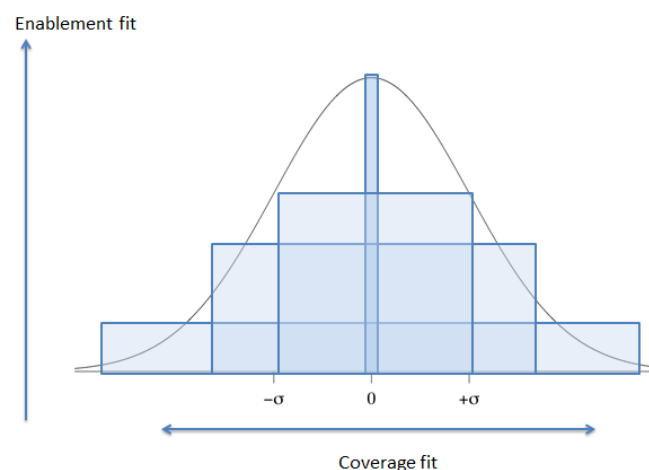


Figure 5: Coverage and enablement fit.

Lastly, after the controlled experiments and subsequent activities have been completed the artifacts are finalized and made ready for publication.

The artifacts created during this thesis project are evaluated and validated by performing controlled experiments with case study data (Hevner et al., 2004). During the experiment, the techniques and method are applied to a sample set containing BPMN process models. The controlled experiments are used to determine the suitability and quality of the techniques and method. A controlled experiment provides the means to control the environment in which the experiment takes place, thus eliminating any confounding variables. A disadvantage of a controlled experiment is the lack of a real-world situation. To counter this, real case study data from different sources is used. Based on the results of the controlled experiments the artifacts are refined and suggestions for future research are formulated.

2 Literature review

As has been explained in section 1.1.1 of this document, the authors of this document recognize a trend in the BPM research field towards constraint-based formalisms. This thesis is a continuation of this trend, since it is an attempt to bridge from traditional formalisms to constraint-based formalisms. In order to get a full grasp on the subject of this thesis a description on flow-based formalisms and constraint-based formalisms is provided below. Furthermore, these formalisms will be placed into context by starting with a concise history of BPM. After that, a description of the techniques used for transformation is given.

2.1 Business Process Management

Academics and businesses alike have put decades of effort in improving business process models. Multiple definitions, such as Business Process Redesign (BPR), Workflow Management and Business Process Management have been used to describe this practice (Ko, Lee, & Lee, 2009). Business Process Redesign can be seen as the predecessor of BPM (Ko et al., 2009). As such, Influential work like Davenport (1993) and Hammer & Champy (1993) have laid the foundation for modern day BPM. In the early days of BPR, the notion was that existing processes should be obliterated and processes should be created from the ground up (Hammer, 1990). Over the years, this notion has been altered to an evolutionary view on business process change (Davenport & Stoddard, 1994; Stoddard & Jarvenpaa, 1995). Additionally, a shift has been made from ad hoc process implementations to methods supporting continuous process improvements, such as described in Harrington & Harrington (1995) and Weske, van der Aalst, & Verbeek (2004).

Closely related to the domain of BPM is Business Rules Management (BRM). In BRM, business constraints are used to restrict or define the business operations. BRM is a declarative approach to capturing these kind of constraints (Goedertier & Vanthienen, 2007). A difference with BPM is that pure sequence related constraints are preferably not captured in business constraints, but are left implicit instead. During the coexistence of BPM and BRM, researchers have promoted and attempted to link the domains together (Kovacic, 2004; Zoet, Versendaal, Ravesteyn, & Welke, 2011). We consider declarative approaches to BPM, such as described by Pesic & Van der Aalst (2006), van der Aalst et al. (2005) and van Grondelle & Gulpers (2011) to be an outcome of this combination.

Whilst the fields of BPR and BPM have gone through many changes, the design and creation of process models has remained a pivotal part. During the existence of BPR and BPM an abundance of process modeling formalisms have been created (Pesic & van der Aalst, 2006; Peterson, 1981; van der Aalst & ter Hofstede, 2005; van der Aalst et al., 2005; Van Grondelle & Gulpers, 2011; Weske, 2007; White, 2004). In this thesis, these formalisms are distinguished into flow-based formalisms and constraint-based formalisms. flow-based formalisms can be defined as formalisms where the focus is on the sequence of activities, which results in procedural process models as described by Goedertier and Vanthienen (2007). Constraint-based process models are recognized by explicit business constraints and the execution scenario is inferred from these business constraints. Table 1 contains the properties of flow-based formalisms and constraint-based formalisms as described by Goedertier and Vanthienen (2007).

	Flow-based formalisms	Constraint-based formalisms
Business constraints	Implicit	Explicit
Execution scenario	Explicit	Implicit
Execution mechanism	State-driven	Goal-driven
Modality	What must	What must, ought and can
Rule enforcement	Procedural (what, when, how)	Declarative (what)
Communication	Explicit (how)	Implicit (what)

Table 1: Properties of flow-based and constraint-based formalisms. Adapted from Goedertier and Vanthienen (2007).

Business constraints, the execution scenario and to some extent the communication property provide the most noticeable differences between the two types of formalisms. These properties have a direct effect on how a process model should be modeled. For instance, in a flow-based formalism communication can be represented by an activity that creates an artifact and a consecutive activity which sends that artifact. In a constraint-based formalism, this can be modeled by only creating the artifact. The remaining three properties are less obvious when comparing process models, because they affect how a process model should be executed. For example, the execution mechanism is not represented by a specific concept in the process model. Rather, the difference in execution mechanism lies in how a supporting application is supposed to handle the process model. The transformation in this research focuses on the properties that have an effect on the process model.

In the following paragraphs several flow-based formalisms and constraint-based formalisms are described.

2.2 Flow-based formalisms

The first process modeling formalism described in this section is Petri Net (Peterson, 1981). Petri Net is a state based formalism originally intended for modeling systems, but has been applied to other fields of study, including business processes (van der Aalst, 1998a). The main benefit for using Petri Nets to model business processes is that it has been defined extensively in formal semantics, it is state-based instead of event based and many analyzing techniques are available (van der Aalst, 1998b).

Places, transitions, arcs and tokens are used as concepts in Petri Net. A place has a connection to another place through a transition. A transition can have an incoming arc (relation) and an outgoing arc. Tokens are used to indicate which transition is executed next. Petri Nets have several limitations in modeling processes (van der Aalst & ter Hofstede, 2005). Three problems that arise in the modeling of processes are a lack of support for multiple instances in sub-processes, lack of support for advanced synchronization patterns and cumbersome modeling caused by the fact that tokens can only be accessed locally (van der Aalst & ter Hofstede, 2005). Yet Another Workflow Language (YAWL) was created to overcome these limitations.

YAWL is based on the concepts of Petri Net, but has its own formal semantics. It incorporates additional concepts, such as multiple instances, a distinction between atomic and composite tasks and joins and token removals (van der Aalst & ter Hofstede, 2005). With the addition of these concepts, it is able to overcome the problems when using Petri Net for business process modeling.

A commonly use process modeling formalism is BPMN (Mendling, Recker, & Reijers, 2011; Recker, 2010; White, 2004; Zur Muehlen & Recker, 2008). BPMN provides a graphical notation for modeling

business processes. Unlike Petri net and YAWL, BPMN does not have any generally accepted formal semantics (Wohed et al., 2006). An example of a BPMN model can be seen in Figure 6.

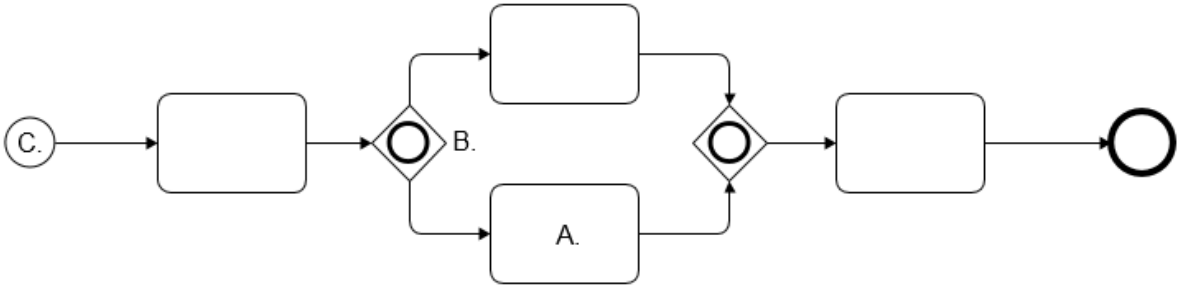


Figure 6: Example of a BPMN process model.

The major components of the BPMN formalisms are activities (A), gateways (B), events (C), exceptions, pools and lanes.

Wohed et al. (2006) provide an overview of the functionalities of more recent formalisms, which can be viewed in Table 2. According to this overview, BPMN currently offers most functionality when modeling workflow patterns. This has been confirmed by research that compared different formalisms using the Bunge-Wand-Weber framework (Recker et al., 2005).

	BPMN	UML 2.0	BPEL	Oracle BPEL PM		BPMN	UML 2.0	BPEL	Oracle BPEL PM
Basic control-flow					11. Implicit termination	+	+	+	+
1. Sequence	+	+	+	+	Multiple instances patterns				
2. Parallel split	+	+	+	+	12. MI without synchronization	+	+	+	+
3. Synchronization	+	+	+	+	13. MI with a priori Design Time Knowledge	+	+	+	+
4. Exclusive choice	+	+	+	+	14. MI with a priori runtime knowledge	+	+	-	+
5. Simple merge	+	+	+	+	15. MI without a priori runtime knowledge	-	-	-	+/-
Advanced Synchronization					State-based patterns				
6. Multiple choice	+	+	+	+	16. Deferred choice	+	+	+	+
7. Synchronization merge	+/-	-	+	+	17. Interleaved parallel routing	+/-	-	+/-	-
8. Multiple merge	+	+	-	-	18. Milestone	-	-	-	+/-
9. Discriminator	+/-	+	-	-	Cancellation patterns				
Structural patterns					19. Cancel activity	+	+	+	+/-
10. Arbitrary cycles	+	+	-	-	20. Cancel Case	+	+	+	+

Table 2: An overview of support for workflow patterns by several flow-based formalisms. Adapted from Wohed et al. (2006).

Examples of other flow-based formalisms for modeling business processes are Colored Petri Net, Event Driven Process Chains, Interaction Flow diagrams, Workflow Nets and Graph based Workflow Language (Weske, 2007).

Several constraint-based formalisms are described in the next section.

2.3 Constraint-based formalisms

Multiple declarative formalisms have been developed in recent research (Pesic & van der Aalst, 2006; van der Aalst et al., 2005; Van Grondelle & Gulpers, 2011). The first formalism discussed in this section is ConDec (Pesic & van der Aalst, 2006). It is based on Petri Net and replaces the states and arcs with constraints, which are based on linear temporal logic (Clarke, Grumberg, & Peled, 1999). These constraints enable the formalism to be declarative. Tokens are replaced with existence constraints, which are depicted on top of the transition concepts.

Another declarative approach to process modeling is the case handling approach by van der Aalst et al. (2005). In this formalism a case is used as the central concept. A case can have activities and data objects. Activities are performed by actors who belong to a certain role. Data objects are accessed by forms, which in turn can be accessed through activities. This formalism results in models where activities are linked with roles, data objects and forms. Business constraints are represented by roles, data object requirements and data object restrictions.

The last declarative formalism discussed in this section is developed at Be Informed and published by van Grondelle & Gulpers (2011). The name of this formalism has recently been changed to Declarative Process Modeling Notation (DPMN). It is based around activities that are performed during a case. Activities can have multiple pre- or post-conditions, such as roles, decisions, objects or artifacts. Pre-conditions serve as a requirement before performing an activity. Post-conditions serve as consequences of activities.

The business constraints are represented by roles, time limits, decisions, artifacts and objects. For example, an activity has the pre-condition "*requires available*" document. This way, no sequentially has to be modeled between activities. This is modeled implicitly by using pre- and post-conditions. DPMN offers the largest amount of concepts compared to the other constraint-based formalisms discussed in this thesis. A DPMN process model DPMN can be viewed in Figure 7.

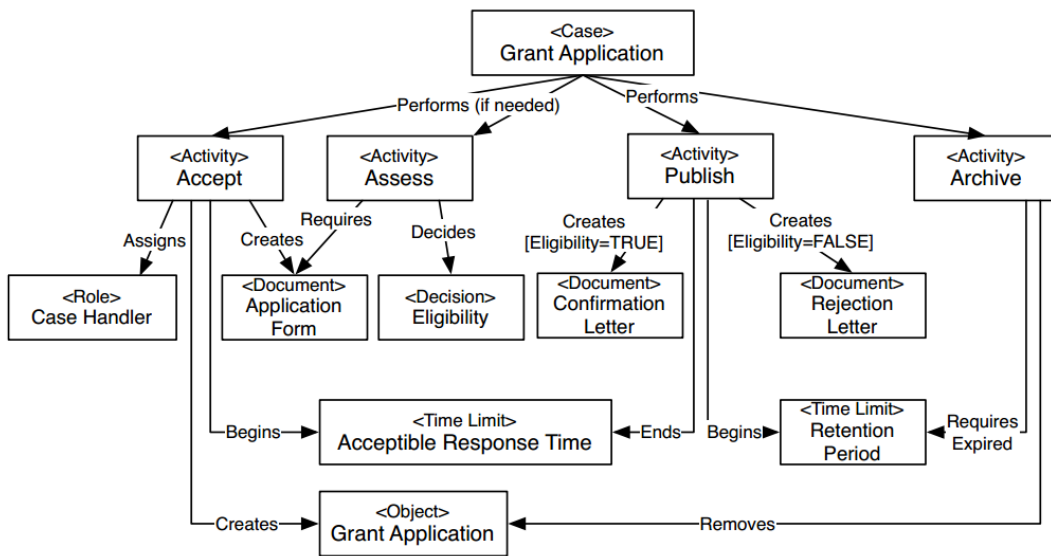


Figure 7: Example of the Declarative Process Modeling Notation. Adapted from Van Grondelle & Gulpers (2011).

Examples of other constraint-based formalisms not explained in this section are Penelope and EM-BRA²CE (Goedertier et al., 2007; Goedertier & Vanthienen, 2006).

To demonstrate the differences between procedural and declarative process models, a mapping is provided between the concepts of the two types of formalisms in Figure 8.

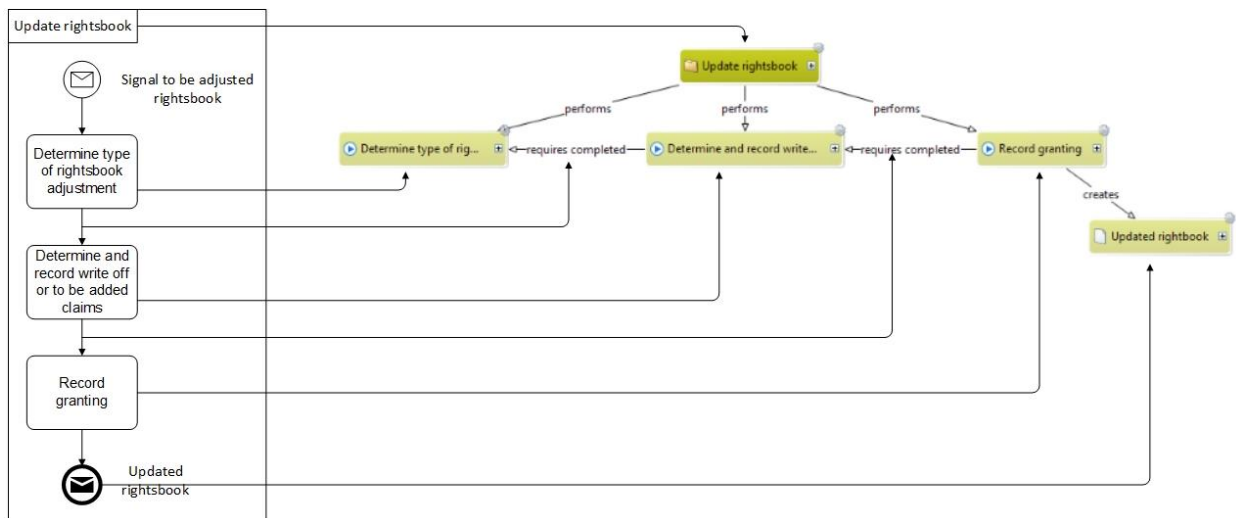


Figure 8: Mapping between concepts of a procedural and declarative formalism.

As can be seen in the figure above, the process name in BPMN corresponds to a case concept in DPMN. The start event in BPMN is omitted from DPMN. Activities in BPMN remain activities in DPMN. Sequence flow between activities in BPMN corresponds to a “requires completed” constraint in DPMN. Lastly, a message end event corresponds to a document in DPMN.

In the next paragraph several flow-based and constraint-based formalisms are characterized based on the declarative properties proposed by Goedertier and Vanthienen (2007).

2.4 Declarative properties mapped to process modeling formalisms

In the previous paragraph several flow-based and constraint-based formalisms are presented and related literature on these formalisms was provided. In this paragraph, an analysis of the similarities and differences between several of these formalisms is performed using the declarative properties proposed by Goedertier and Vanthienen (2007). The declarative properties are described in Table 3 (Goedertier & Vanthienen, 2007).

Property	Description
Execution scenario	The execution scenario is related to how a process is executed. Among others, an execution scenario describes which activities to perform at what time. In flow-based formalisms this is modeled explicitly. All possible execution scenarios are modeled at design time. In constraint-based formalisms this is modeled implicitly. The constraints are modeled at design time and the execution scenario is inferred from these constraints at runtime.
Business constraints	Business constraints are modeled implicitly in flow-based process models. A procedural process model is modeled to stay within the given business constraints, but the business constraints are not modeled itself. In constraint-based process models, business constraints are modeled explicitly.
Execution mechanism	Flow-based process models are state-driven. In other words, they are executed without regard of any goal to be achieved. Constraint-based formalisms are goal-oriented. A goal is defined and the formalism determines which activities and other artifacts are required for achieving this goal.
Modality	Modality is related to a distinction in concepts that <i>must</i> , <i>ought</i> or <i>can</i> be performed. In flow-based formalisms activities must be performed, since the flow of activities prescribes a certain order in which to perform them. In constraint-based formalisms an activity <i>must</i> , <i>can</i> or <i>ought</i> be performed. For example, at runtime certain activities are mandatory for achieving the goal. An actor can decide to perform a non-mandatory activity if this does not violate the constraints.
Rule enforcement	Rule enforcement is related to <i>what</i> , <i>when</i> and <i>how</i> rules are executed. In flow-based formalisms, rules are modeled within the flow (For example: an XOR waypoint is modeled within the order of activities). As such, it is prescribed <i>what</i> , <i>when</i> and <i>how</i> rules are enforced. In constraint-based formalisms, rules are modeled explicitly and enforcement of these rules is not influenced by any order of activities. As such, only <i>what</i> is enforced in constraint-based formalisms.
Communication	In flow-based formalisms communication is often modeled explicitly. For example, “ <i>send invoice</i> ” is an explicit way of modeling communication. In constraint-based formalisms events and artifacts can be perceived by actors. As such, explicit communication is not necessary.

Table 3: Declarative properties proposed by Goedertier & Vanthienen (2007).

As can be read from Table 4, Petri Net, BPMN, YAWL, Colored Petri Net, Event Driven Process Chains, Interaction Flow Diagrams, workflow nets and graph based workflow language (which are considered procedural formalisms) do not feature any of the declarative properties. Furthermore, DPMN, EM-BRA²CE and Penelope feature all declarative properties.

The Case Handling Paradigm features several declarative properties. This formalism maintains procedural order between activities at design-time, but deviations are allowed with the right permissions at run-time. Additional constraints can be placed on data objects, requiring the user to provide the data before continuing with the process. Therefore, this formalism has an explicit execution scenario and explicit business constraints. Furthermore, the formalism is not goal oriented, since it does not infer what needs to be done based on the goal of the process. Properties four and five are featured by this formalism. Users are able to skip, redo or perform activities based on their permissions. Lastly, communication is explicit in this formalism.

Condec features the first and second property by using Linear Temporal Logic (LTL) as constraints. These constraints are used to infer the order of activities. Goal orientation is not featured, since it is not possible to specify a goal in this formalism. The fourth and fifth properties are only partly featured, since the formalism partly provides declarative modality and rule enforcement. Lastly, the sixth property is not featured by Condec, since communication can only be explicitly modeled in activities (instead of e.g. a document concept).

Formalisms	1)Execution scenario Implicit?	2)Business constraints Explicit?	3)Execution mechanism goal driven?	4)Modality	5)Rule enforcement	6)Communication Implicit?
Petri Net	No	No	No	Must	What/when/how	No
BPMN	No	No	No	Must	What/when/how	No
YAWL	No	No	No	Must	What/when/how	No
Colored Petri Net	No	No	No	Must	What/when/how	No
Event driven process chains	No	No	No	Must	What/when/how	No
Interaction flow diagrams	No	No	No	Must	What/when/how	No
Workflow nets	No	No	No	Must	What/when/how	No
Graph based workflow language	No	No	No	Must	What/when/how	No
Case handling paradigm	No	Yes	No	Must/can/ought	What	No
Condec	Yes	Yes	No	Must/can	What/when	No
DPMN	Yes	Yes	Yes	Must/can/ought	What	Yes
Penelope	Yes	Yes	Yes	Must/can/ought	What	Yes
EM-BRA ² CE	Yes	Yes	Yes	Must/can/ought	What	Yes

Table 4: Declarative Characteristics of Several Process Modeling Formalisms.

2.5 Quality characteristics in different types of formalisms

A small amount of research has been done in comparing the different types of formalisms according to quality characteristics in software (Zeist & Hendriks, 1996). In a recent paper, Pichler, Weber, Zugal, Pinggera and Reijers (2012) mention that claims and discussions about the presumed advantages of the types of formalism are manifold, but empirical validation of these claims is lacking. As such, Table 5 contains research with claims about a type of process modeling formalism backed up by empirical validation. It should be noted that most of this research used a single formalism of each type to perform the study, thus the results may not be representative for the complete spectrum of formalisms for each paradigm.

Characteristic	Procedural formalisms	Declarative formalisms
Functionality		Declarative formalisms are more flexible than contemporary approaches (Schonenberg, Mans, Russel, Mulyar, &

		van der Aalst, 2008; van der Aalst, Pesic, & Schonenberg, 2009).
Reliability		
Usability	Procedural formalisms provide better understandability (Pichler et al., 2012).	
Efficiency	Implementation time for workflow management is less than case management (Weber, Mutschler, & Reichert, 2010).	
Maintainability		Test cases significantly improve maintainability of declarative approaches (Zugal, Pinggera, & Weber, 2011).
Portability		

Table 5: Quality characteristics in different types of process modeling formalisms.

Schonenberg et al. (2008) propose a taxonomy of flexibility for process modeling languages that includes four types of flexibility. Out of the process modeling formalisms compared in their research, Condec supports most types of flexibility. In contrast, procedural formalisms are found to provide better understandability (Pichler et al., 2012). This research was performed with BPMN and Condec as procedural and declarative formalisms respectively. It is unclear if other formalisms provide different results. In a comparison between the implementation time of case handling and a traditional workflow approach, the traditional approach provides better results (Mutschler, Weber, & Reichert, 2008). Lastly, it was determined that test cases significantly improve the maintainability of declarative process formalisms (Zugal et al., 2011).

The next paragraph describes which techniques are commonly used for model transformation or can be used for this purpose.

2.6 Transformation techniques

In the next sections the techniques used for model transformation are explained. An introduction is given for each technique and it is explained how the technique can be applied for model transformations. The selection criteria for the techniques are their ability to consider business constraints, execution scenario and communication properties (as described in Table 4) during the transformation.

2.6.1 Graph transformation

The first transformation technique discussed in this thesis is based on graph transformation theory. In this section, a brief introduction to the concepts involved in graph transformation is given. For a complete and full definition we refer to Ehrig & Ehrig (2006) and Ehrig, Prange, & Taentzer (2004). A basic definition of a graph based model transformation can be written as: $MT : VL_S \rightarrow VL_T$, where MT is a model transformation function using productions for the transformation, VL_S is a source visual language and VL_T a target visual language (H. Ehrig & Ehrig, 2006). More concrete, a meta-model is created, which consists of an attributed type graph ATG . Within ATG , the meta-model of the source visual language ATG_S and the meta-model of the target visual language ATG_T are included, together with reference concepts and relations that are required for the model transformation (H. Ehrig & Ehrig, 2006). The transformation is performed by a graph transformation system GTS, which

is defined as $GTS = (ATG, prod)$ (H. Ehrig & Ehrig, 2006). Where *prod* are productions and can be considered as the transformation rules (H. Ehrig & Ehrig, 2006). When applying this technique, a visualization of the meta-model can be used to depict the target and source models and how their concepts are related (Varró, Varró, & Pataricza, 2002). In the meta-model, graph nodes are represented by rectangular shapes and node attributes with their data type are placed below the graph node name. The graph nodes are related to each other by arrows with an inscription. Reference nodes are depicted by a square box with a dotted line. An example can be viewed in Figure 9, where a formalism consisting of tasks and sequences is transformed into Petri Net.

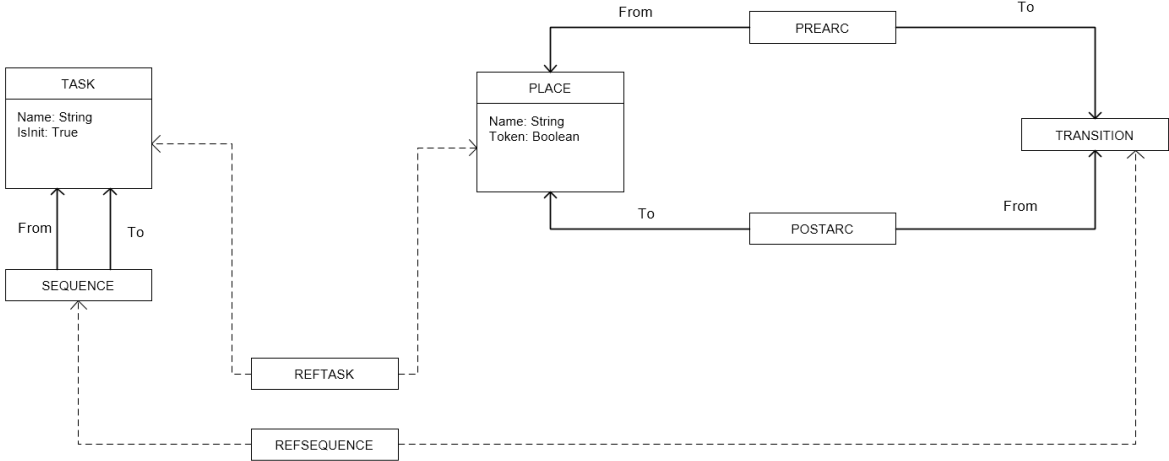


Figure 9: Example of a graph transformation.

The productions used in this transformation are REFTASK and REFSEQUENCE. REFTASK is used to transform tasks in the source model to places in the target model (Figure 10). Subsequently, REFSEQUENCE is used to insert transitions and arcs between places that are connected tasks in the source model (Figure 11).

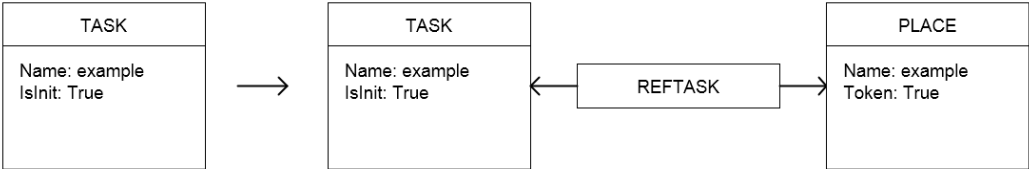


Figure 10: Example of production "REFTASK".

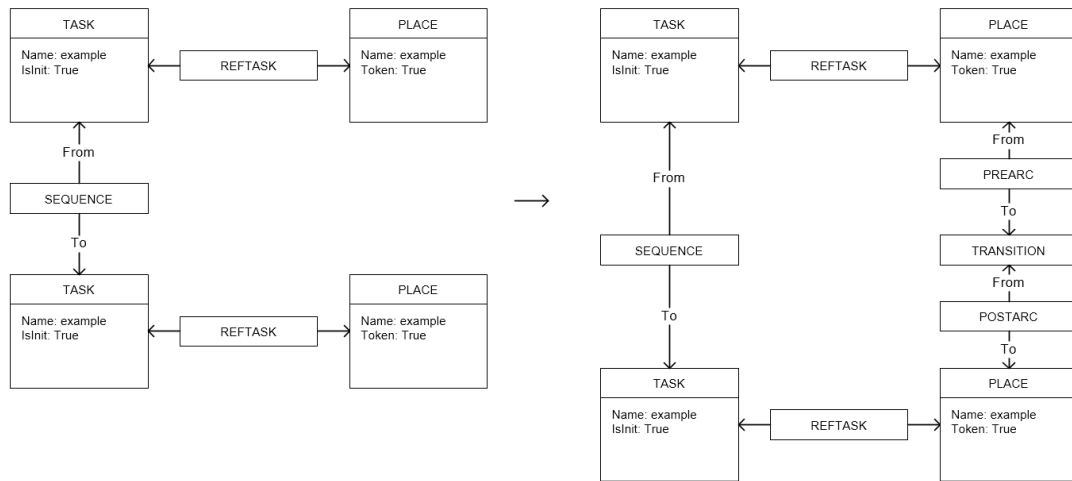


Figure 11: Example of production "REFSEQUENCE".

2.6.2 Transforming Well-formed BPMN

The second syntactical transformation technique is the Well-formed BPMN transformation technique. With this technique, predetermined well-formed clusters of concepts of the source formalism are transformed into predetermined clusters of concepts in the target formalism (Dijkman, Dumas, & Ouyang, 2008; Ouyang, Dumas, Ter Hofstede, & van der Aalst, 2006). In their paper, Dijkman et al. (2008) recognize the following BPMN concepts (Figure 12):

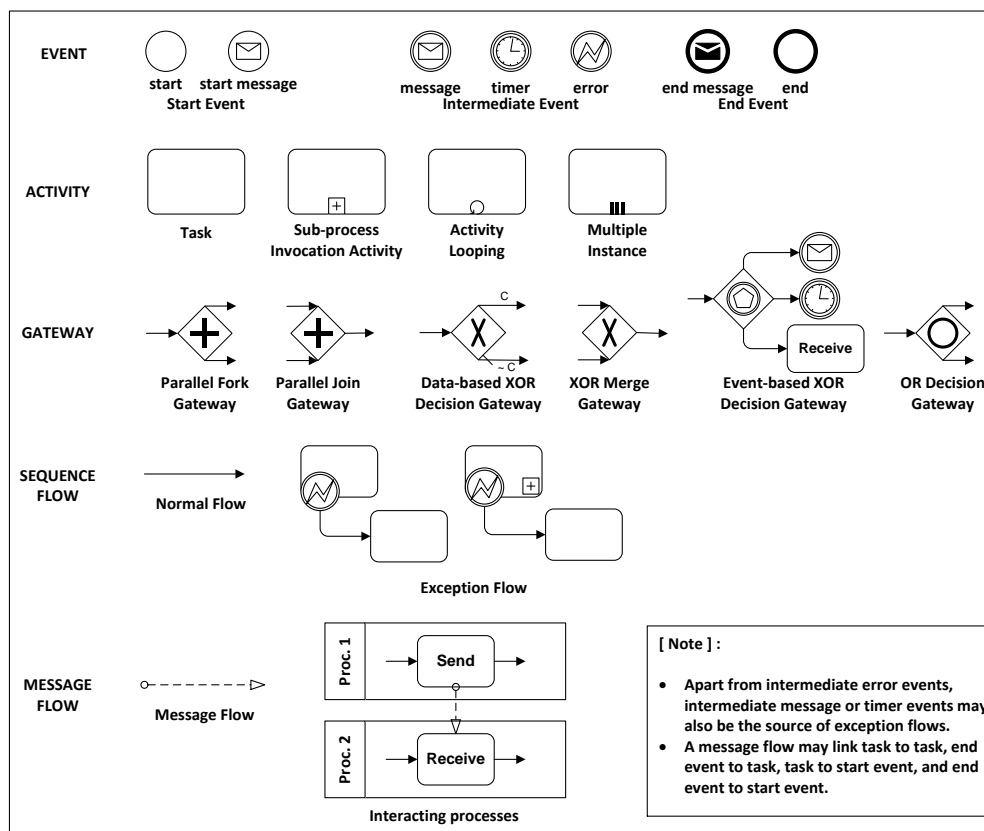


Figure 12: Well-formed BPMN clusters.

In addition to the standard BPMN concepts, Dijkman et al. (2008) compose a definition of Well-formed BPMN process model using the following rules: 1) Start and error events have only one

outgoing flow; 2) End events have only one incoming flow; 3) activities and intermediate events have one incoming and one outgoing flow; 4) split gateways have one incoming flow and multiple outgoing flows; 5) merge gateways have multiple incoming flows and one outgoing flow. In the context of this research these rules can be considered as preconditions for BPMN models. As such, Well-formed BPMN process models are ready for transformation, while non-well-formed BPMN process models need to be remodeled to conform to these rules.

Using the well-formed BPMN rules, the concepts are expanded by also including a small part of their context. For example, an AND split gateway is expanded with one incoming flow from an unknown concept and multiple outgoing flows to unknown concepts. For each BPMN cluster there is a corresponding figure in the target modeling formalism (when possible and applicable). Three examples of these transformations are shown in Table 6.


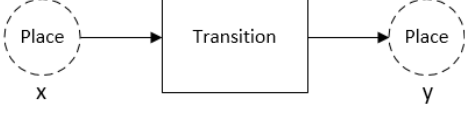
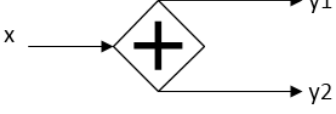
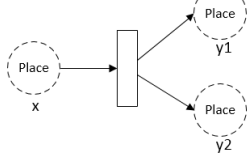
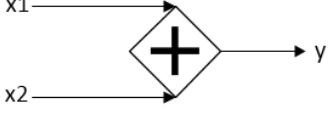
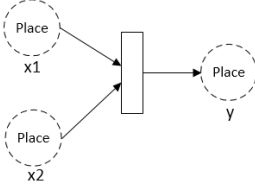
Source cluster	Target cluster
	
	
	

Table 6: Example cluster transformations.

The transformations in the table above are from BPMN to Petri Net. This technique will be adjusted to make it suitable for transformations from BPMN to the DPMN.

2.6.3 Related cluster pair technique

The related cluster pair technique is based on finding similar nodes or clusters based on pre-defined query nodes or clusters (Niemann et al., 2012). This technique can be used to retrieve pre-defined clusters and replace them with a template target cluster. This technique uses a combination of similarity measurements to determine if a node or cluster is similar to a query cluster. A cluster is part of a Process Model Graph, which is defined by Niemann, Siebenhaar, Schulte and Steinmetz (2012) as:

Definition 1: (Process Model Graph). Let G be a graph $G = (V, E)$, Λ be a set of labels and Θ be a set of types. A process model graph P is a directed, weakly connected graph defined as tuple $P = (V, E, \lambda, \tau, \alpha)$, where:

- V is a finite set of nodes

- $E \subseteq (V \times V)$ is a finite set of edges
- λ is a labeling function: $\lambda : (V \cup E) \rightarrow \Lambda$ that assigns labels to nodes and edges
- $\tau : (V \cup E) \rightarrow \Theta$ assigns types to nodes and edges, and
- $\alpha : (V \cup E) \rightarrow (A \rightarrow \Lambda)$ assigns attributes to nodes and edges, where A is a set of attributes that are assigned labels.

In particular, the sets A , Θ and Λ all include \in (the NULL element).

As can be read from the definition, a process model graph consists of different types of nodes and edges with labels assigned to them. Based on this definition, a cluster is defined by Niemann et al. (2012) as:

Definition 2: (Cluster). Let $P = (V_p, E_p, \lambda_p, \tau_p, \alpha_p)$ be a PMG and Θ a set of cluster types. A cluster L in P is a connected subgraph $(V, E, \lambda, \tau, \alpha)$ such that:

- $V \subseteq V_p$ (nodes) and $E \subseteq E_p$ (edges)
- $S = (V, E)$ is a SESE region:
 - $|\{(u, v) \mid (u, v) \in E_p \wedge v \notin V\}| = 1 \wedge$
 - $|\{(u, v) \mid (u, v) \in E_p \wedge u \notin V\}| = 1$
- $\lambda = \lambda_p, \tau = \tau_p$ and $\alpha = \alpha_p$ are functions as in Def. 1
- the function $t : \{L\} \rightarrow \Theta$ assigns a type to the cluster

The set Θ contains single nodes, node, node sequences, node or cluster loop and gateway constructs.

A cluster has only one incoming edge and one outgoing edge and contains one or more nodes that are not a gateway. Consequently, each node that is not a gateway is a cluster and clusters may contain other clusters. Furthermore, a related cluster pair is defined by Niemann et al. (2012) as follows:

Definition 3: (Related Cluster Pair). Let L_1 and L_2 be clusters $L_1 = (V_1, E_1, \lambda_1, \tau_1, \alpha_1)$ and $L_2 = (V_2, E_2, \lambda_2, \tau_2, \alpha_2)$, and $V_Q \subseteq (V_1 \times V_2)$ a set of nodes. A related cluster pair Q is defined as $Q = (V_Q, \text{sim}^{\text{node}}, t)$ where sim^{node} is a node similarity function and t a similarity threshold $t \in \mathbb{R}$ with $0 \leq t \leq 1$.

For all $(x, y) \in V_Q$, the following conditions apply:

- $\text{sim}^{\text{node}}(\lambda_1(x), \lambda_2(y)) \geq t$ (similarity of nodes based on labels)
- $\exists!(v, w) \in V_Q : v = x \vee w = y$ (unique node assignment)
- $\tau_1(x) = \tau_2(y)$ (equality of node types)
- $t_{L_1}(L_1) \sim^{\text{ctSim}} t_{L_2}(L_2)$, (similarity of cluster types)

Where \sim^{ctSim} is a binary relation specifying the similarity of the cluster types contained in Θ (Def. 2)

A cluster is related to another cluster when each node of cluster L_1 is uniquely assigned to a node in cluster L_2 based on label similarity and is above the similarity threshold. Furthermore, the assigned nodes have to be of the same type and the type of the clusters has to be similar. The threshold for assigning similarity between two nodes is determined to be 0.18 for the measurements applied in this research, based on cross-validation of the results (Niemann et al., 2012).

Initially, clusters are related to each other in their smallest form (one node). Then, clusters are merged if they conform to the condition that if node A and B are adjacent in model 1, node A and B also have to be adjacent in model 2. Also, the resulting clusters have to be related to each other. How to determine similarity is explained in the next section.

Measurement

Niemann et al. (2012) have performed experiments with several similarity measurement techniques. According to their results, the Levenshtein distance metric in combination with a word synonym

metric aggregated by the Monge Elkan similarity metric provides the most precision when applied for process model retrieval and comparison (Niemann et al., 2012). Therefore, these measurements are applied for this technique. Notice that both metric are applied on the labels of the concepts. All calculations for measuring similarity are made in MS Excel. The functions created for these calculations can be found in Appendix C.

The Levenshtein distance metric ($lev(l_1, l_2)$) is a string based metric which calculates the edit distance between two strings (Levenshtein, 1966). In other words, the amount of editing steps required to change label l_1 into label l_2 . For example, the Levenshtein distance between “make” and “making” is three, since one letter has to be changed and two have to be added. Niemann et al. (2012) apply the metric as follows:

$$sim^{lev}(l_1, l_2) = 1 - \frac{lev(l_1, l_2)}{\max(|l_1|, |l_2|)} \quad (1)$$

Where $|l_1|$ and $|l_2|$ are used to indicate the length of the string. Going back to the example, applying $sim^{lev}(l_1, l_2)$ to “make” and “making” results in a similarity measurement of 0.5. However, measuring similarity based on string edit distance has limitations when it comes to similar words with a different meaning. For example, the labels “plane” and “planet” have a similarity value of 0.83 but are semantically not related. To account for this limitation, the Levenshtein distance metric is used in combination with a semantic similarity measurement. The semantic similarity of two words is determined by the following metric (Niemann et al., 2012):

$$sim_{0.5}^{ws}(w_1, w_2) = \begin{cases} 1 & \text{if } w_1, w_2 \text{ are identical words} \\ 0.5 & \text{if } \exists \text{ synset } S \text{ with } w_1, w_2 \in S \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The sim_a^{ws} formula compares two given words to each other. When words are identical a value of 1 is returned. When a word is contained in the same synset in WordNet (Miller, 1995)(a synset is a set of semantically identical words) a value of 0.5 is returned. Otherwise, the metric returns the value of 0. The returning values are aggregated with the Monge Elkan similarity metric, which is adapted for $sim_{0.5}^{ws}(w_1, w_2)$ and defined in Niemann et al. (2012) as:

$$Sim^{me}(A, B, sim_{0.5}^{ws}) = \frac{1}{|A|} \sum_{i=1}^{|A|} \max_{j=1}^{|B|} sim_{0.5}^{ws}(A_i, B_j) \quad (3)$$

This aggregating metric can be applied to any word similarity metric by switching $sim_{0.5}^{ws}$ for another formula. For this metric, both labels are separated into single words. For each word in label A it returns the highest value of $sim_{0.5}^{ws}(w_1, w_2)$ for each word in label B . For example, the labels “create bill” and “create invoice” have a similarity value of 0.75.

According to Niemann et al. (2012), the weight of the string based metric (Levenshtein distance) should be 0.1 (Niemann et al., 2012). Accordingly, the weight of the semantic metric is 0.9. All this taken into account, the weighted similarity value of “create bill” and “create invoice” is 0.73.

2.6.4 Measuring semantic similarity between process models

The previous techniques perform similarity measurements on individual concepts or clusters in a process model. The goal of this technique is to recognize a similar process model by looking at the similarity of the process model concept labels to a set of predetermined labels, which are used as queries (M. Ehrig, Koschmider, & Oberweis, 2007). When the process model is similar to the query model, it is transformed according to a template model in the target constraint-based formalism. The following paragraph describes which metrics are used in this technique.

Measurement

Measuring semantic similarity is done using a similar approach as the related cluster pair technique, but on the level of complete process models. All calculations for this technique are also made in MS Excel. The functions created for these calculations can be found in Appendix C. This technique determines process model similarity based on a syntactic, linguistic and structural similarity measurement. The syntactic similarity metric is based on the Levenshtein distance metric and is defined in M. Ehrig et al. (2007) as:

$$sim_{syn}(c_1, c_2) = \max(0, \frac{\min(|c_1|, |c_2|) - lev(c_1, c_2)}{\min(|c_1|, |c_2|)}) \quad (4)$$

Where c_1 and c_2 represent a concept label from model 1 and 2 respectively. Instead of dividing the result by the longest process label, as was done in the related cluster pair technique, it is now divided by the smallest process label. Consequently, $sim_{syn}(c_1, c_2)$ for the labels “make” and “making” is 0.25. The semantic similarity metric based on linguistics is defined in M. Ehrig et al. (2007) as follows:

$$sim_{ling}(c_1, c_2) = \frac{f(S)}{\max(|\eta(c_1)|, |\eta(c_2)|)} \quad (5)$$

Where $\eta(c)$ is defined as the set of senses returned from WordNet (Miller, 1995) given the concept label c and $f(S)$ is defined in M. Ehrig et al. (2007) as:

$$f(S) = \begin{cases} 1 & \text{iff } \eta(c_1) \cap \eta(c_2) \neq \emptyset \\ 0 & \text{iff } \eta(c_1) \cap \eta(c_2) = \emptyset \end{cases} \quad (6)$$

The linguistic metric measures the semantic similarity between two process labels based on the intersection of senses for the words in the labels. When a label has multiple words, the words that have a sim_{syn} of 1 are omitted from the metric, because (according to the authors of the technique) this does not make sense from a linguistics perspective. For example, a similarity value would be returned for the labels “make friends” and “make car”. However, these labels are not semantically related. Therefore, a linguistic similarity measurement between “friends” and “car” is preferred.

The next metric considers the structural context of a concept. The structural context consists of several elements that are related to the original activity concept, such as data attributes, data values and succeeding activities. This metric measures the syntactic and semantic similarity between the context elements from concept c_1 and concept c_2 . Each context element is assigned a specific

similarity measurement and weight. M. Ehrig et al. (2007) provide a table and measurements for Petri Net process models supplemented with attributes and attribute values (Table 7):

Comparing	Context element	Measure	Weight
Places	Names	sim_{syn} / sim_{ling}	0.2
	Attributes	sim_{syn} / sim_{ling}	0.4
	Value	sim_{syn}	0.1
	Successor	sim_{syn} / sim_{ling}	0.3
Attribute	Names	sim_{syn} / sim_{ling}	0.2
	Sibling attribute	sim_{syn} / sim_{ling}	0.4
	Values	sim_{syn}	0.4
Value	Names	sim_{syn} / sim_{ling}	0.2
	Attribute	sim_{ling}	0.4
	Values reference	sim_{ling}	0.4
Transition	Name	sim_{syn} / sim_{ling}	0.2
	Successor	sim_{syn}	0.8

Table 7: similarity measurements and weights for extended Petri Net models.

The measurements for the context elements are aggregated as follows (M. Ehrig et al., 2007):

$$sim_{str} = \frac{\sum_{i=1}^{|c_1|} \max_{j=1}^{|c_2|} (w_{k_i} sim_{k_i}(c_1, c_2))}{\sum_{i=1}^{|c_1|} w_{k_i}} \quad (7)$$

Where $|c_1|$ and $|c_2|$ denote the amount of context concepts for concept c_1 and c_2 respectively. Furthermore, sim_{k_i} denotes the similarity measurement for that specific context element and w_{k_i} denotes the weight assigned for that similarity measurement. As such, for each context element in c_1 the maximum similarity value is returned for each context element in c_2 , where the similarity value is calculated based on the assigned similarity metric and assigned weight. The resulting summation of similarity values is divided by the sum of weights assigned to each context element in c_1 .

After each similarity metric has been calculated for c_1 and c_2 the results are combined and weighted into a single similarity value with the following calculation (M. Ehrig et al., 2007):

$$sim_{com}(c_1, c_2) = \frac{w_{c_{syn}} sim_{syn}(c_1, c_2) + w_{c_{ling}} sim_{ling}(c_1, c_2) + w_{c_{str}} sim_{str}(c_1, c_2)}{w_{c_{syn}} + w_{c_{ling}} + w_{c_{str}}} \quad (8)$$

Where $w_{c_{syn}}$, $w_{c_{ling}}$ and $w_{c_{str}}$ are the weights assigned to each similarity metric. The weights can be set by users manually or determined with for example a machine-learning approach. Lastly, the Monge Elkan metric is used to aggregate the results for the complete sets of concepts C_1 and C_2 (M. Ehrig et al., 2007):

$$sim_{SBPM}(C_1, C_2, sim_{com}) = \frac{1}{|C_1|} \sum_{i=1}^{|C_1|} \max_{j=1}^{|C_2|} (sim_{com}(c_{1_i}, c_{2_j})) \quad (9)$$

2.7 Explication techniques

In the next sections the techniques used for constraint explication are explained. An introduction is given for each technique and it is explained how the technique can be applied for constraint explication. The selection criteria for the techniques are their ability to consider business constraints, execution scenario and communication properties (as described in Table 4) during the explication of constraints.

2.7.1 Lexical analysis of activity labels

The lexical analysis of activity labels technique can be used to interpret the original intention of an activity is. The goal of this technique is to tag activities based on the verb used in the activity label. Activities are chosen for interpretation, since they are one of the central concepts of process models (Mendling et al., 2011; Recker, 2010). To determine which tag an activity label should get, a verb classification model can be used. An example of an already existing verb classifying model can be found in Mendling, Recker and Reijers (2011). In their work 25 verb classifications were created based on different taxonomies, which are able to classify 95% of the activities in the SAP reference process models.

For each verb tag a corresponding structure in the target formalism is available, thus making transformation based on lexical analysis possible.

2.7.2 Classifying process fragments using machine learning

With this technique it is determined how machine learning can aid in classifying process fragments. To determine which machine learning algorithm supports the purpose of this technique most precise, experiments with the most used classifying algorithms as categorized in Wu et al. (2007) are performed. The C4.5, kNN, Naïve Bayes and CART algorithms are explained shortly in the sections below.

In the context of this research these algorithms can be used to determine if a process concept is of a certain class based on activity labels as attributes (e.g.: if it creates or sends a document). When it is classified as such, that process concept is tagged and can be transformed, expanded or omitted, depending on the specific rule for that tag.

C4.5

The C4.5 algorithm can be used to create decision trees in order to classify instances (Quinlan, 1993). The tree is constructed based on a pre-classified set of instances. In order to determine the structure of the tree it looks at what attribute out of the set of attributes has the highest normalized information gain. The attribute with the highest normalized information gain value is used to branch the decision tree. This process is repeated until the decision tree is completed. The algorithm stops

creating branches when a stopping criterion is met. For example, a total amount of branches has been created or the information gain is below a certain threshold. After the tree is constructed, it is pruned based on estimated error to reduce complexity and overfitting.

For example, given the instances in Table 8, attribute 1 would be used for the first branch, since it provides the highest normalized information gain.

Attribute 1	Attribute 2	Class
True	True	True
True	False	True
False	True	False
False	False	False

Table 8: Example of a C4.5 dataset.

kNN

The kNN (k-nearest neighbor) algorithm classifies instances based on their place in a feature space (Fix & Hodges, 1951; Tan, Steinbach, & Kumar, 2006). The place of an instance is determined based on the attribute values of that instance. Which class the instances are given depends on the classes of the nearest neighbors of that place, where the amount of neighbors for a classification can vary and the class is determined by the most common class of the neighbors.

Take the example of Figure 13. Assume the class of instance 1 and 2 are determined based on the three nearest neighbors. Instance 1 would be classified as blue, because all three neighbors are blue. Instance 2 is classified as red, because two out of three neighbors are red. If the class of an instance would be based on the five nearest neighbors, both instances would be classified as blue.

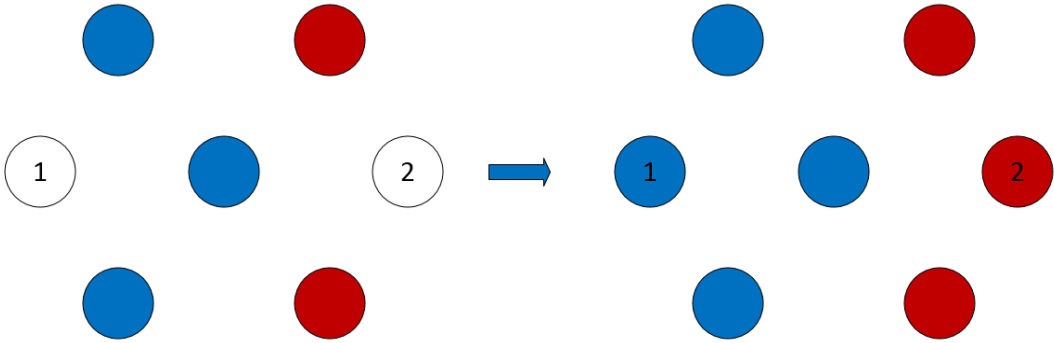


Figure 13: Instances in a feature space.

Naïve Bayes

The Naïve Bayes classifier is one of the most common techniques for data mining (Witten, Frank, & Hall, 2011). It is used to classify an instance based on a given sample set of instances (Tan et al., 2006). As input, the technique requires a classified set of instances and a collection of variables relating to the instances. Based on the variables and if the instance belongs to a class, the technique is able to calculate the probability that a new instance belongs to a class. Take the example of classifying a car as either a family car or sports car (Table 9). There is a sample set which denotes which class the car is for each instance, how many horsepower the car has, if the car has four-wheel

drive and the price of the car. Based on this information, the probability that instance x is a sports car or family car can be calculated.

#	Price	Four-wheel drive	Horsepower	Type of car
1	25,000	No	250	Family car
2	100,000	Yes	550	Sports car
3	20,000	No	200	Family car
4	150,000	Yes	650	Sports car
n
x	125,000	Yes	500	?

Table 9: Example of a Naïve Bayes classifier sample set.

CART

CART is another algorithm for constructing decision trees (Breiman, Friedman, Olshen, & Stone, 1984). The main differences with the C4.5 algorithm are that it does not apply a criterion to stop branching the tree (Wu et al., 2007). It allows for the maximum possible size of the tree and uses pruning afterwards to reduce the size and complexity of the tree. Furthermore, it applies a different criterion for determining the next branch and uses cost-complexity for pruning the tree instead of the estimated error (Wu et al., 2007). Lastly, a large part of the CART algorithm is dedicated to dealing with missing values (Wu et al., 2007).

The next chapter describes how these techniques are adjusted for the experiments and how their performance is measured. To perform the techniques, a source flow-based formalism and target constraint-based formalism have to be selected. Different methods of analysis conclude that BPMN offers the most functionality compared to other flow-based formalisms. Furthermore, it is identified that DPMN, Penelope and EM-BRA²CE conform to all of the given declarative properties. Due to practical reasons, DPMN was chosen as the target formalism.

3 Data collection

In this chapter the process of data collection is described. To determine the coverage and enablement fit, several measurements are done. Coverage fit for the techniques is measured by the percentage of transformed or correctly interpreted process model (concepts). Coverage fit for the method is measured by the amount of syntactically correct transformed process models. Enablement fit for the method is measured with an expert review of the transformed process models. To collect the coverage and enablement fit data required for the final method, several iterations through the research framework are required (Figure 14). During the first iteration, the techniques (described in the previous chapter) are adapted to make them suitable for transformation from BPMN to DPMN or to explicate constraints from transformed process models. Each technique is then evaluated using a controlled experiment. After each technique is evaluated, the second iteration starts. During this iteration, a method is created using the adapted techniques from the previous iteration. The method is evaluated in a second controlled experiment with different case study data. The second iteration ends with finetuning the method based on the results of the controlled experiment. Then, during the third iteration, the method is used to transform several BPMN process models from the second set of case study data. The transformed process models are evaluated by DPMN modeling experts, resulting in an enablement fit rating (Strong & Volkoff, 2010). Lastly, final improvements are made based on the feedback received during the expert review.

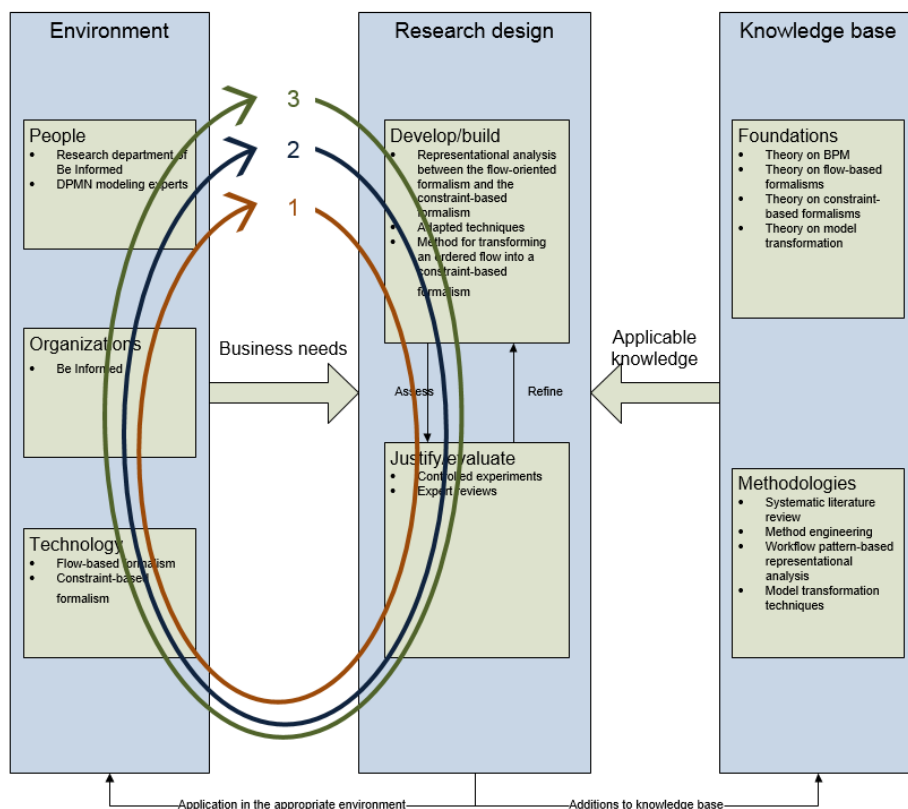


Figure 14: Several iterations through the research framework.

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 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.

Because there are several iterations through the research framework multiple units of analysis are distinguished. The units of analysis are the adopted techniques and the method.

3.1 Unit of analysis 1: Techniques

The techniques mentioned in the previous chapter are one of the units of analysis of this research. Each technique is adjusted in order to make it suitable for a transformation from BPMN to DPMN or to identify the intention of an activity based on the label and add a tag to the concept. An explanation on how each technique was adjusted can be found in the sections below. For each technique it needs to be determined how suitable it is for providing a syntactical transformation or interpret the intention of the activity. Suitability is measured in coverage fit, as mentioned in section 1.3. More specifically, coverage fit for the techniques used in a syntactical transformation 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 techniques that identify activity intentions and apply tags, coverage fit is measured by the percentage of correct and logical classifications.

Because some of the techniques are similar to each other (for example: Well-formed- and Graph transformation are both used for syntactic transformation) the technique learning might occur (Juristo & Moreno, 2010, p. 116). This effect occurs when performing multiple experiments with the same technique. When this occurs, the outcome of the experiments might be affected by the experience gained from performing the first technique. The object learning effect occurs when experimenters are experienced with the problem to be dealt with (in this case the sample set), because multiple techniques are applied to the same problem. To avoid these effects in this research, the well-formed and lexical analysis transformation techniques were adjusted and used in experiments by another researcher.

3.2 Unit of analysis 2: Method

After each technique has been adjusted and evaluated, they are put into a method that combines all techniques. This method is the second unit of analysis. The combined techniques all result in transformed concepts or clusters of concepts and tags on concepts. As such, a single concept can have multiple tags. The priority of tags needs to be determined during the evaluation of the method. The method is evaluated by performing transformation on a second set of process models from another source. During this controlled experiment, coverage fit is measured by the percentage of syntactically correct transformations. Based on the results of the evaluation, the method is improved by changing the order of priority for the tags. The final method is used to transform another set of process models. The transformed process models are rated by three DPMN modeling experts in order to measure the enablement fit (Strong & Volkoff, 2010).

4 Data analysis

This chapter contains the results of the iterations described in the previous chapter. Paragraph 4.1 described the adjustments and results of the techniques. Paragraph 4.2 described how the method is constructed and the performance of the method in transforming process models. Lastly, in paragraph 4.3 the evaluation by DPMN Modeling experts and the final method are described.

4.1 Techniques

In the sections below the adjustments to the techniques are explained. Furthermore, the performance of the techniques on the first sample set is described.

4.1.1 Graph transformation

To successfully transform the source models into the target models, productions need to be created which specify how to transform each source concept into a target concept. The first 59 source process models were used as a learning set to determine the productions. For each source model in the learning set all productions required to transform the models were recorded. This resulted in a set of productions required for the transformation. The remaining 44 models were used to verify the productions.

Upon visual inspection of the source models a few minor errors were fixed, such as the addition or redirection of a sequence flow, adding a label to an XOR waypoint and adding a missing merge waypoint. The error fixes did not change the intention of the process model. Six source models were excluded, because they contain severe syntactic errors. Two examples of the productions used during the transformation can be read in Table 10. The remainder of the productions can be found in Appendix E.

Production	Description
REFPROCESS	For each process model a case concept is created in the target DPMN model. The name of the process model corresponds to the name of the case concept.
REFACTIVITY	An activity in BPMN corresponds to an activity in DPMN. In order to maintain the given order of activities. If an activity is preceded by another activity in the BPMN source model, a <i>“requires completed”</i> condition is placed between these activities in the DPMN model accordingly.

Table 10: Example productions for the graph transformation technique.

When accounting for the six process models omitted from this technique, a total of 97 process models were eligible for transformation. From this group 24 models were not able to be transformed, resulting in a coverage fit of 0.75 for the applicable models. When accounting for all process models, the coverage fit is 0.71. The reasons for not being able to transform a model can be viewed in Table 11 (several process models have multiple problems).

Reason	Amount
Loop	15
Intermediate timer events	6
Non True/false XOR	4
Sub-process after AND merge	1
Complex XOR structures	4

Table 11: Reasons for not being able to transform a model.

Figure 15 depicts an example of a transformation. The arrows between models represent a specific production and show the source and result of a production. The start message event is not transformed into a concept in DPMN, but is omitted from the target model.

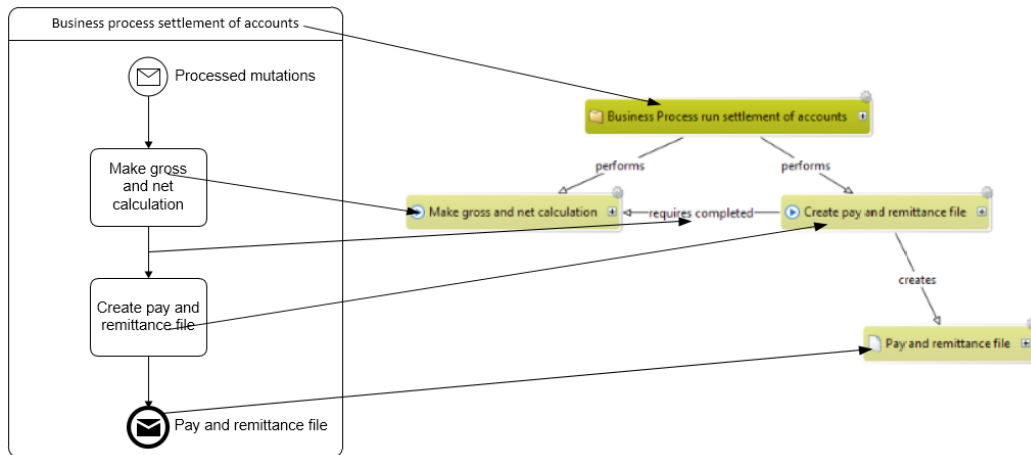


Figure 15: Example of a transformation.

4.1.2 Well-formed BPMN transformation

This section describes the adaption of the well-formed BPMN Technique in order to make it suitable for transformation between BPMN and DPMN. The source clusters used in Dijkman et al. (2008) were also as source clusters in this research. The target clusters were changed for clusters in DPMN. An excerpt of the resulting list of transformation rules can be seen in the table below (Table 12). The full set of transformation rules can be viewed in Appendix D.

Source cluster	Target cluster
<p>Message E1</p>	
<p>Task T</p>	

Table 12: Transformation rules for the well-formed transformation technique.

These transformation rules were used to transform the sample set. Fifty-two of the source models were discarded, because they contain multiple start or end events. While having multiple start or end events is allowed, it is unclear how this should be interpreted (Dijkman et al., 2008). Of the remaining 51 process models, 33 process models were transformed correctly and 18 contained syntactical errors. The cause of these errors is summarized in Table 13 (several models have multiple causes).

Reason	Amount
Loop	5
Split gateway without preceding activity	4
XOR split immediately followed by an XOR merge	12
Non True/ False XOR gateway	3
Complex XOR structure	3

Table 13: Errors in the Well-formed transformation technique.

As such, the coverage fit of this technique for the process models eligible for transformation is 0.64. When accounting for all 103 process models, the coverage fit amounts to 0.32.

4.1.3 Related cluster pair transformation

In this section it is explained how the related cluster pair technique, described in paragraph 2.6.3, is used for the transformation from BPMN to DPMN. Before a measurement can be done an adjustment to the formal definition of a cluster (Definition 2) needs to be made in order to make it compatible with the sample BPMN models. The definition of Θ is expanded to include events:

The set Θ contains single nodes, node, node sequences, node or cluster loop, gateway constructs and events.

In order to determine the similarity between clusters a similarity measurement for each activity label needs be performed first. For measuring the similarity between activity labels, the formulas $sim^{lev}(l_1, l_2)$ and $Sim^{me}(A, B, sim_{0.5}^{ws})$ are used with a weight ratio of 0.1 and 0.9 respectively, as explained in paragraph 2.6.3. The performance of the cluster retrieval with is measured with Mean Average Precision:

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m} \sum_{k=1}^{m_j} Precision(R_{jk})$$

Where Q is a set of queries, m is the set of relevant activity labels and R_{jk} is the ranked list of search results. $Precision(R_{jk})$ measures precision by dividing j with the total amount of retrieved search results up until m_j .

The suitability of this technique for cluster retrieval (and transformation when a similar cluster is found) is validated by using a similar way as in Dijkman, Dumas, Dongen, Reina and Mendling (2011) and Niemann et al. (2012). Out of the 103 process models, six process models were randomly selected. From these process models the activity labels were translated to English in order to make it compatible with WordNet (Miller, 1995). Furthermore, the labels were stemmed in a similar way as described in Dijkman et al. (2011) to make the words more easily comparable. From each of these process models a cluster was randomly selected to be used as query. Different types of changes were

made to pairs of clusters. For the first two clusters (1 and 2) no change was made. For the second pair of clusters (3 and 4) the activity labels were changed without changing the meaning of the labels. For the last pair of clusters (5 and 6) the order of activities was changed.

The first cluster consists of two consecutive activities, where the first activity is related to creating an artifact and the second is related to sending that artifact. The second cluster has a rework loop, where the first activity is related to verifying data, followed by an XOR waypoint that is connected to an activity with a label related to explaining differences, correcting errors or requesting extra data (Query 3 and Query 4).

When the first cluster is found (left side of Figure 16) it can be transformed into a corresponding cluster in DPMN (right side of Figure 16). Documents are a separate concept in DPMN and are also assumed to be available for each person involved. Therefore, they do not have to be send separately. This results in a transformation of two activities in a BPMN model into one document concept in a DPMN model.

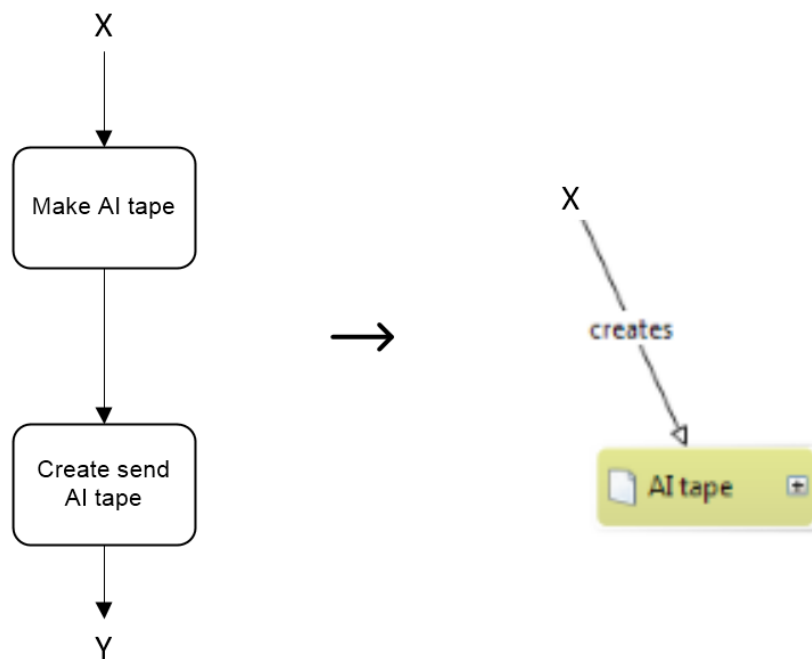


Figure 16: Transformation of cluster 1.

For finding the first cluster mentioned in the previous section two queries were formulated (Query 1 and Query 2). The query for the first activity is “make AI tape” with the synonyms “produce”, “create”, “recording” and “taping”. The first query results in a precision and similarity value for the corresponding activity of 1.

Query 1:		Precision:
Word(s):	Make AI tape	1
synonym(s):	Produce, create, recording, taping	

Table 14: Query 1 for the related cluster pair technique.

The second query is the retrieval of the activity labels that sends the AI tape. The words “create send AI tape” were used for the query with the accompanied synonyms “produce”, “create”, “mail”,

“post”, “recording” and “taping”. This corresponds to the words used for the label of this activity. Again, a precision and similarity value of 1 was achieved.

Query 2:		Precision
Word(s)	Create send AI tape	1
synonym(s)	Produce, create, mail, post, recording, taping	

Table 15: Query 2 for the related cluster pair technique.

For the other clusters similar results were achieved. The results can be viewed in Table 16.

Query	Cluster	Precision	Similarity value
3	2	1	1
4	2	1	1
5	3	1	0.64
6	3	0.5	0.47
7	4	1	0.75
8	4	1	0.73
9	4	1	1
10	5	1	1
11	5	1	1
12	6	1	1
13	6	0.5	1

Table 16: Results of the cluster retrieval experiment.

However, the goal of this technique is not finding one specific cluster. Rather, it is aimed at finding all clusters that share a similar intention. Therefore another experiment was performed in which the queries were generalized and an attempt was made to find multiple clusters with the same intention. For this demonstration, two types of clusters are searched for. The first cluster consists of two consecutive activities, where the first activity is related to creating an artifact and the second is related to sending that artifact (Query 14 and Query 15, on page 37 and 38). The second cluster is a rework loop, where the first activity is related to verifying data, followed by an XOR waypoint that is connected to an activity with a label related to explaining differences, correcting errors or requesting extra data (Query 16 and Query 17, on page 38).

When the first cluster is found (left side of Figure 16) it can be transformed into a target cluster in DPMN (right side of Figure 16). Documents are a separate concept in DPMN and are assumed to be available for each person involved and therefore they do not need to be send separately. This results in a transformation of two activities in a BPMN model into one document concept in a DPMN model.

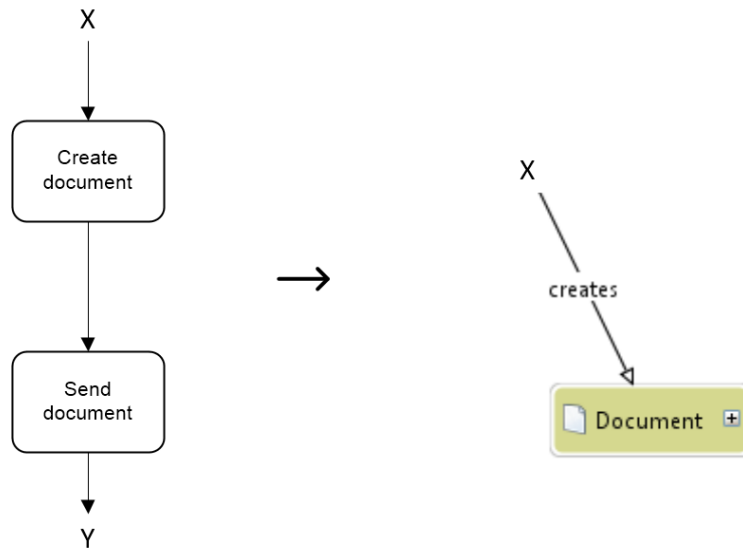


Figure 17: Example of a related cluster pair transformation.

The second cluster is depicted on the left side of Figure 18. When a cluster with a similar intention is found it is transformed into the target cluster as shown on the right side of Figure 18. Exception loops are omitted from DPMN models and only the verification activity is transformed to the target model.

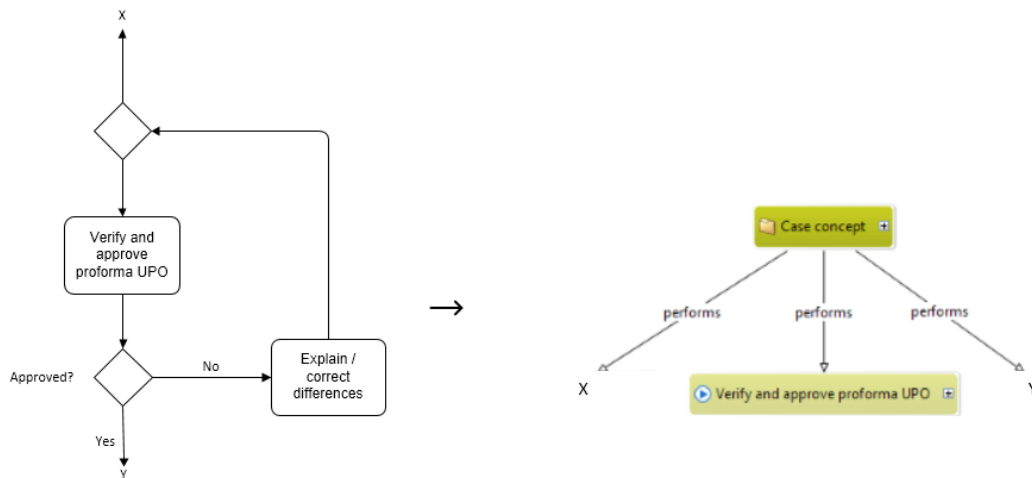


Figure 18: Another example of a related cluster pair transformation.

For finding the first cluster mentioned in the previous section two queries were formulated (Query 14 and Query 15). The query for the first activity is “make” with the synonyms “produce” and “create”. Upon visual inspection of the sample set, 23 activities that create an artifact were marked. This query resulted in a mean average precision of 0.74.

Query 14:	Activities that create an artifact	Precision:
Word(s):	make	0.74
synonym(s):	Produce, create	n=23

Table 17: Query 1 for the related cluster pair technique.

The second query is the retrieval of all activity labels that send an artifact. The word “send” was used for the query with the accompanied synonyms “mail” and “post”. During visual inspection of the

activity labels, eight activities that send an artifact were marked. The query resulted in a mean average precision of 1, meaning the first eight search results were all of the marked activities.

Query 15:	Activities that send an artifact	Precision
Word(s)	send	1
synonym(s)	Mail, post	n=8

Table 18: Query 2 for the related cluster pair technique.

Based on a visual inspection it was determined that there is one cluster in the sample set that has two consecutive activities, where the first activity is related to creating an artifact and the second one is related to sending an artifact. The cluster has an average similarity value of 0.93 with the queries. This indicates that it is a related cluster pair, since both labels have a similarity degree higher than the threshold (0.18).

The second cluster is searched for with the queries 16 and 17. Upon visual inspection, there are six clusters that conform to the description given in the text above. The first query (Query 16) is concerned with finding the activity that verifies the completeness and correctness of an artifact. The query used is “verify” and results in a precision of 0.21.

Query 16:	Verifying data	Precision:
Word(s)	verify	0.21
synonym(s)	n.a.	n=6

Table 19: Query 3 for the related cluster pair technique.

Query 17 is used to find a rework task, because the artifact was not verified to be complete or correct. A rework task can either be an explanation, a correction or a request. Each of the words was treated as a separate query in order to get the best result out of the sim^{lev} equation. The mean average precision of Query 4 is 0.27.

Query 17:	Explanation, correction or request	Precision:
Word(s)	Explain, correct, request	0.27
synonym(s)	Explicate, rectify, right	n=6

Table 20: Query 4 for the related cluster pair technique.

Out of the six clusters determined before performing Query 16 and Query 17, four clusters were found with an average similarity value of 0.93. For the remaining two clusters only one of the activity labels returned a similarity value above the threshold, thus not conforming to the definition of a related cluster.

Concluding, the mean average precision for all queries is 0.55. Of the seven clusters marked by visual inspection, five were recognized as related clusters according to the definition.

4.1.4 Measuring semantic similarity between process models

Eight randomly selected process models from the sample set were to test if a semantic similarity measurement is suitable for process model retrieval. From these process models, the activity labels were translated to English in order to make it compatible with WordNet (Miller, 1995). Furthermore, the labels were stemmed in a similar way as described in Dijkman et al. (2011) to make the words more easily comparable. The process labels of these models are used as queries and adjusted in a

similar way as in Dijkman et al. (2011) and Niemann et al. (2012). The first two process models (1 and 2) are left unchanged, the labels of the second pair of process models (3 and 4) are changed without changing the meaning of the labels, half of the labels were removed from the query for the third pair of process models (5 and 6) and lastly, the order of activities was changed for the last pair of process models (7 and 8). No query with changes to the connector type was used since BPMN uses sequence flow as control flow mechanism from and to activities.

Before a similarity measurement can be performed the context elements measured by sim_{str} need to be adjusted, because the source models do not contain all context elements mentioned in the example in paragraph 2.6.4. For this experiment the name of the succeeding activity was considered as the only context element. When a succeeding activity exists, sim_{str} has a weight ratio of 0.2, sim_{syn} a weight ratio of 0.3 and sim_{ling} a weight ratio of 0.5. If no succeeding activity exists, sim_{syn} gets assigned a weight value of 0.4 and sim_{ling} a weight value of 0.6. Furthermore, the authors of this technique do not specify which weight ratio to use when calculating sim_{str} . In this experiment the highest value returned from sim_{syn} and sim_{ling} is used as value for sim_{str} .

Comparing	Context element	Measure	Weight
Activities	Name	sim_{syn} / sim_{ling}	0.8
	Successor	sim_{syn} / sim_{ling}	0.2

Table 21: Context elements considered for this experiment.

The resulting similarity values for each query can be seen in Table 22. Surprisingly, the first two queries did not score a similarity value of 1, which is expected because they were left unchanged. This is caused by the removal of similar words in composed activity labels. Furthermore, query 5 to 8 did not receive the highest similarity value for their corresponding process model. The mean average precision (as explained in paragraph 4.1.3) is 0.64.

Query	Similarity value
1	0.4
2	0.48
3	0.62
4	0.44
5	0.29
6	0.38
7	0.47
8	0.42

Table 22: Similarity values for the randomly selected process models.

4.1.5 Classifying process labels using machine learning

In order to determine the most suitable algorithm for classifying process labels the percentage of correct classifications of the sample set for each algorithm is calculated. The algorithms are executed using the Weka application (Hall et al., 2009). In this experiment, activity labels are used as instances to classify process fragments. The activity labels were translated to English and were stemmed in a similar way as described in Dijkman et al. (2011). To calculate the precision for each algorithm, activities that 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 case study 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. 28 instances were given a True value and 489 were given a False value. The classification of the instances beforehand enables the algorithm to learn if an instance creates or sends a document based on the activity labels. Secondly, all the activity labels were transformed into a vector. In other words, each unique word becomes a separate variable for all instances and gets assigned a nominal value of 1 or 0 depending on the presence of that word in the process label. This resulted in 256 attributes on which the classification is based. Because some algorithms might suffer from a large amount of attributes The correctness percentages of each algorithm for the top 15 attributes ranked by their information gain is also calculated (Witten et al., 2011). Furthermore, each classifier was performed 10 times and in each run 10-fold cross validation was applied. The average percentage of correctly classified instances is shown in Table 23.

	Naïve Bayes	C4.5	CART	kNN
All attributes	94.70	97.64	97.80	96.93
Top 15 attributes	98.61	97.64	97.89	97.72

Table 23: Correctness percentages for the most common classifier algorithms.

The results show that for each technique classifications based on the top 15 attributes perform equal or better than classification based on all attributes. Furthermore, the Naïve Bayes algorithm has the highest average correctness percentage of the classifiers tested (98.61%). This translates to approximately seven classification errors out of 517 process labels. As such, the Naïve Bayes algorithm was used for the machine learning technique in the remainder of this research.

4.1.6 Lexical analysis of activity labels

The first step for this technique is determining the correct classifications for the labels. The process labels in the sample set containing 103 process models were classified with the classifications 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 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 the DPMN. Two examples of the resulting classification scheme can be viewed in Table 24. The remaining classifications can be viewed in Appendix F.

Main verb	Related verbs DPMN	Related verbs Case study data	Related verbs Mendling et al. (2011)
1. To determine	<ul style="list-style-type: none"> • Determine • Assess • Review • Verify • Validate • Decide • Check 	<ul style="list-style-type: none"> • Determine • Assess • Figure out • Verify • Analyze • Compare • Approve • Test 	<ul style="list-style-type: none"> • Assess • Decide
2. To create	<ul style="list-style-type: none"> • Create 	<ul style="list-style-type: none"> • Create 	<ul style="list-style-type: none"> • Create

		<ul style="list-style-type: none"> • Make • Generate 	
--	--	--	--

Table 24: Verb classification scheme.

With this classification scheme, 328 of the 517 labels can be classified. 100 of the labels could not be classified, because they do not contain any verbs. The other 89 labels could not be classified, because the verb could not be placed in any of the classifications. Out of the 328 of classified verbs, 276 of the corresponding structures are logical. This leaves 52 of the classified verbs to have a corresponding structure that does not make sense. As such, the coverage fit for this technique is 0.84. When the not applicable activity labels are also taken into account, the coverage fit amounts to 0.53.

4.1.7 Evaluation of techniques

Looking at the results for each technique, there are advantages and disadvantages to each of them. The Graph and well-formed BPMN transformation techniques offer a high percentage of coverage fit, but do not interpret the process model concepts. This means that they possess the ability to transform a high percentage of models, but are not able to take advantage of the declarative properties of DPMN. The Lexical analysis of activity labels and machine learning techniques are able to classify most of the activities in the sample set and provide a semantic interpretation. However, the interpretation is very broad and limits the specificity of the corresponding structures. The related cluster pair and semantic process similarity techniques have the capability to provide a precise transformation. However, these techniques require domain knowledge and predetermined queries for it to work, causing the coverage fit to be low. The results from the techniques as described in the above section 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. Together with the rules of the Well-formed BPMN and graph transformation techniques, the tagging rules Appendix G will be used in the transformation method.

4.2 Method description

Before the method is explained, some explanation on how the techniques are used in the method needs to be given first. Two approaches for combining the techniques were attempted. During the first approach, all techniques (except the semantic process similarity technique) were used to provide tags for (clusters of) concepts simultaneously. The selection of tags to use for the transformation was based on priority. This approach resulted in several limitations. By selecting one tag for transformation of a cluster (e.g. in the well-formed transformation technique) other valuable tags for a single concept were discarded in the transformation. Furthermore, merging the separately transformed clusters and concepts proved difficult.

For the second approach a distinction is made between transformations and interpretation techniques. To prevent the loss of tags and difficulties when merging, the transformation techniques were performed first with a distinct order in which to perform the techniques. This already results in a process model in DPMN, but does not include any interpretation of the process model concepts. The techniques on the semantic level are then 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. This approach proved to be more successful and was used to evaluate the method.

The method is presented in a Product Deliverable Diagram (PDD), as described in Van de Weerd & Brinkkemper (2008). The PDD can be viewed in the next section. The activity table is located in section 4.2.2 and the concept table can be found in 4.2.3. The transformation results for this method can be found in 4.2.4.

4.2.1 Product Deliverable Diagram

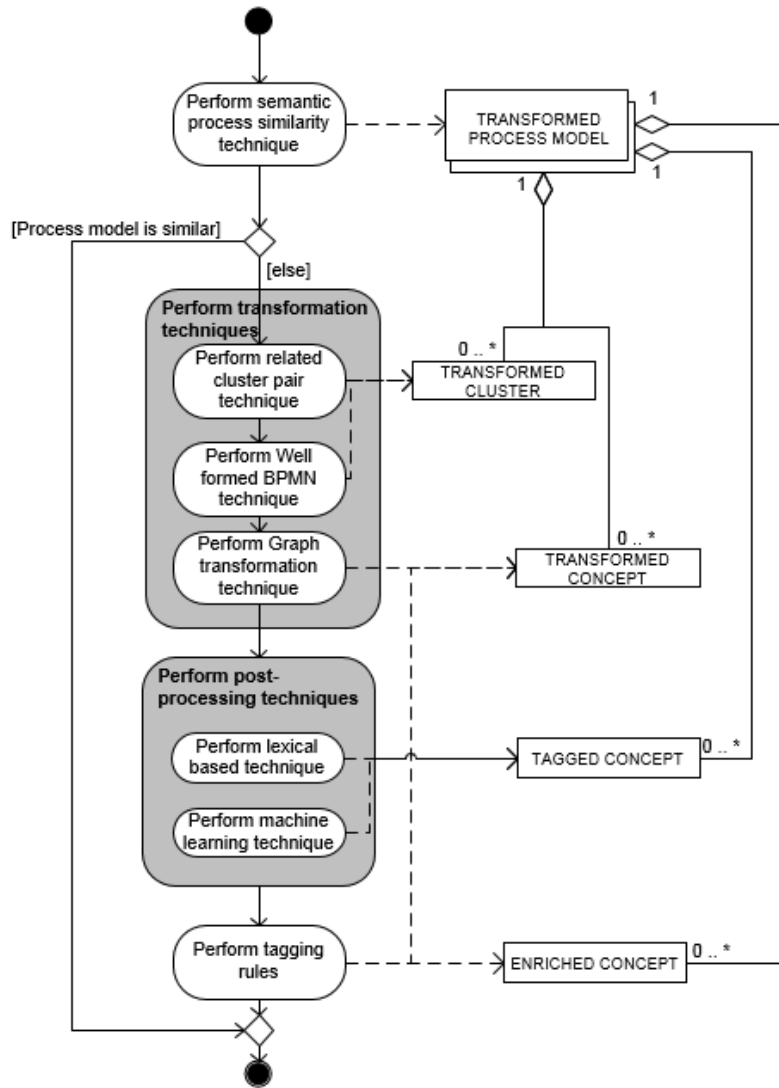


Figure 19: Method for transforming procedural models.

4.2.2 Activity table

Activity name	Sub-activity	Description
Perform semantic process similarity technique		The semantic process similarity technique is performed during this activity. When a process model is found to be similar it is transformed according to a pre-modeled template. The resulting artifact is a TRANSFORMED PROCESS MODEL. Because the model is already completely transformed, no additional activities have to be performed.
Perform transformation techniques	Perform related cluster pair technique	During this activity the Related cluster pair technique is performed. Based on the similarity measurements of the labels in the source model, clusters are

		transformed into a corresponding DPMN cluster. These clusters are stored in the TRANSFORMED CLUSTER concept and are part of a TRANSFORMED PROCESS MODEL.
	Perform well-formed BPMN technique	During this activity the well-formed BPMN technique is performed. This technique results in none, one or more transformed clusters. These are stored in the TRANSFORMED CLUSTER concept.
	Perform graph transformation technique	The graph transformation technique is performed during this activity. This technique transforms individual concepts and stores these tags in the TRANSFORMED CONCEPT concept.
Perform post-processing techniques	Perform lexical based technique	During this activity, tags are applied to activities in the source process model based on their lexical classification. The tags are stored in the TAGGED CONCEPT concept.
	Perform machine learning technique	A Naïve Bayes classifier is used to classify and tag the activities in the source process model, based on their activity labels. The tags are stored in the TAGGED CONCEPT concept.
Perform tagging rules		During the last activity of the method, the tags in assigned to the single concepts in the TRANSFORMED PROCESS MODEL are used to change the model. Based on the priority of the tags, the rule with the highest priority is applied to change, expand or omit the concept.

Table 25: Activity table for the PDD.

4.2.3 Concept table

Concept	Description
TRANSFORMED PROCESS MODEL	This is the final result of the method. A TRANSFORMED PROCESS MODEL consists of concepts and relations as described in the corresponding meta-model (Van Grondelle & Gulpers, 2011). The process is composed of TRANSFORMED CLUSTER(S), TRANSFORMED CONCEPT(S), TAGGED CONCEPT(S) and ENRICHED CONCEPT(S). This concept is either a result from the semantic process similarity technique (M. Ehrig et al., 2007) or a composition of the results of the other techniques and activities.
TRANSFORMED CLUSTER	The well-formed BPMN and related cluster pair techniques are able to transform clusters of concepts (Dijkman et al., 2008; Niemann et al., 2012). As such, results from these techniques are stored in the TRANSFORMED CLUSTER concept.
TRANSFORMED CONCEPT	The graph transformation technique can result in individually transformed concepts (H. Ehrig & Ehrig, 2006). The resulting concepts are stored in the TRANSFORMED CONCEPT concept.
TAGGED CONCEPT	The lexical based and machine learning techniques result in individually tagged concepts. A TAGGED CONCEPT is stored in this concept until it can be used in the “ <i>performs tagging rules</i> ” activity.
ENRICHED CONCEPT	As a result of the “ <i>perform tagging rules</i> ” activity, concepts in the TRANSFORMED PROCESS MODEL are changed, expanded or omitted from the model. These modifications are stored in the ENRICHED CONCEPT concept.

Table 26: Concept table for the PDD.

4.2.4 Transformation results

The method described above and the transformation rules as mentioned in Appendix G were used in the transformation of 10 BPMN process models from a healthcare institution. Based on the resulting transformed models, the priority of the tag rules was changed. Then, the experiment was repeated with the adjusted priority. Out of the 10 process models, 9 could be successfully transformed. One could not be transformed because it contains a loop structure, which causes a deadlock in DPMN. An example of the transformation steps and resulting model is depicted in Figure 20.

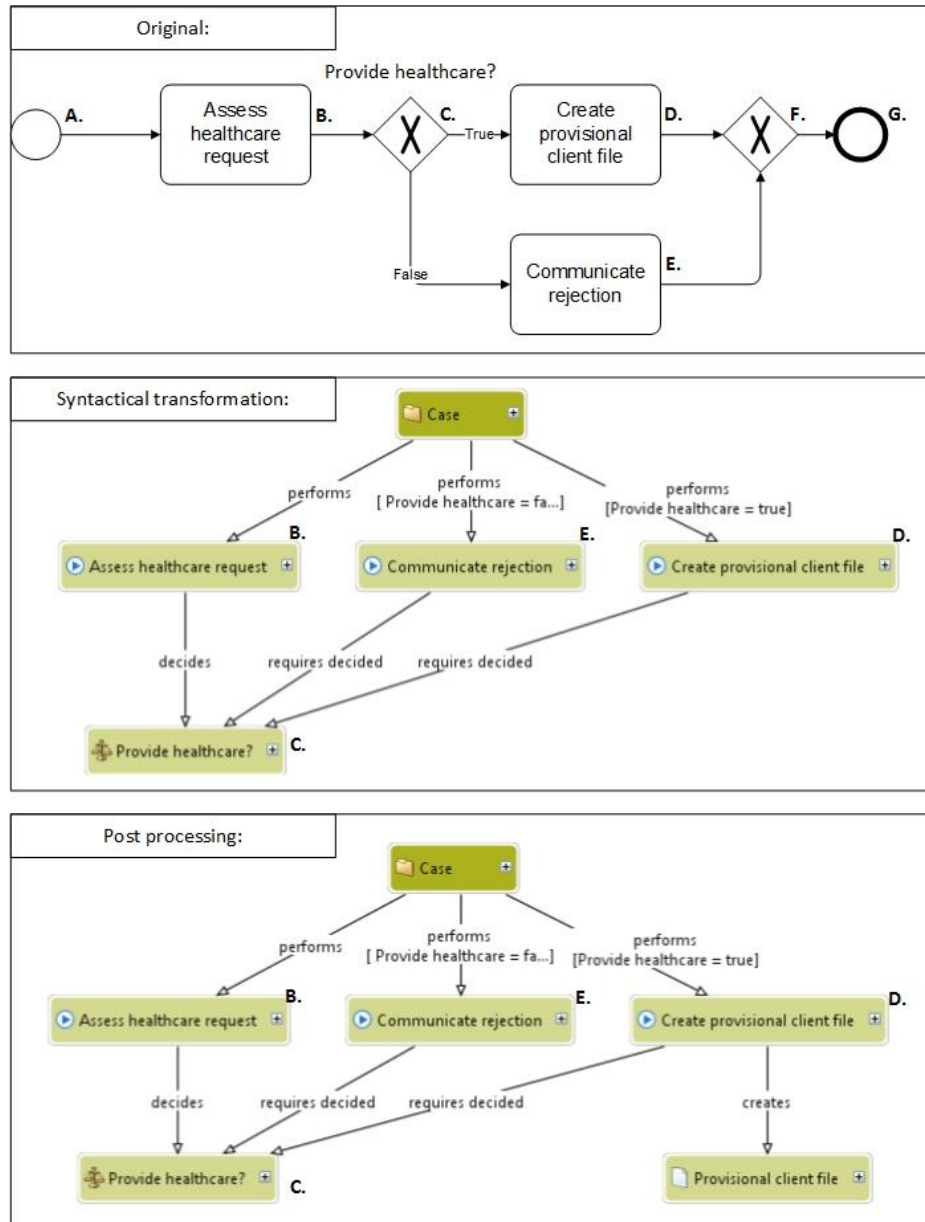


Figure 20: Example of transformation steps and resulting process model.

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. Based on the similarity values resulting from the measurements no related cluster can be found. Therefore, 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 the Data Based decision

cluster (D1) and are transformed to the corresponding DPMN cluster (See Appendix D for the Well-Formed BPMN clusters). 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 determined during the first iteration of this research (Appendix G), the concepts A, F and G are omitted from the target model. Since all concepts have been transformed, the method continues with the next phase. During the next phase, post-processing techniques are performed. The machine learning technique classifies concept B as a “communicate” and “determine” activity concept (Appendix G). The lexical analysis technique tags concept B as a “determine” activity (Appendix G). Concept E is tagged as “determine” and “communicate” activity by the machine learning technique. It is also tagged as “communicate” activity by the lexical analysis technique. The last 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 concepts are tagged, the post processing phase is finished and the tagging rules are applied in the next method activity. The process model 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 already contains a decision concept, no enrichment takes place for this concept. The tag with the highest priority of concept E does not contain any additional concepts. Therefore, concept E is left as is. Lastly, concept D is enriched by adding a “document” concept to the activity.

Accounting for all successfully transformed process models, 20 more concepts were added during the post-processing activity. For example, in Figure 20 the document concept “provisional client file” is added. Upon visual inspection, 9 of the additional concepts were considered illogical. This means that 55% of the added concepts are useful and add value to the model.

4.3 Final method

The final transformed process models from the second iteration were presented to three DPMN modeling experts with at least one year of experience. The experts were asked to rate each model for their enablement fit. On average, the models were rated a 5.6 on a scale from 1 to 10. 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 what activities need to be performed to reach the goal of the process model. Lastly, some activities are included in the target models that do not seem to serve any function. The method is currently not able to distinguish relevant and irrelevant activities. All experts agreed that the syntactical techniques deal with XOR's 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. Lastly, it is currently not possible distinguish relevant and irrelevant activities. Therefore, this will not be attempted during the expert review activity. An example of how the expert review activity would change a process model can be viewed in Figure 21.

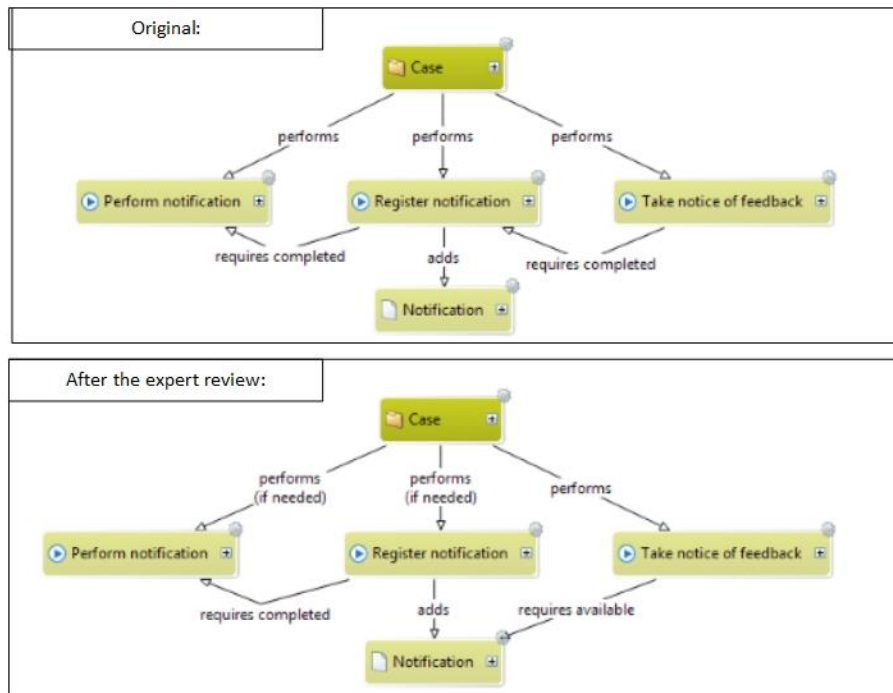


Figure 21: Example of the expert review step.

When the “perform expert review” activity is added to the method, it looks as follows:

4.3.1 Final Product Deliverable Diagram

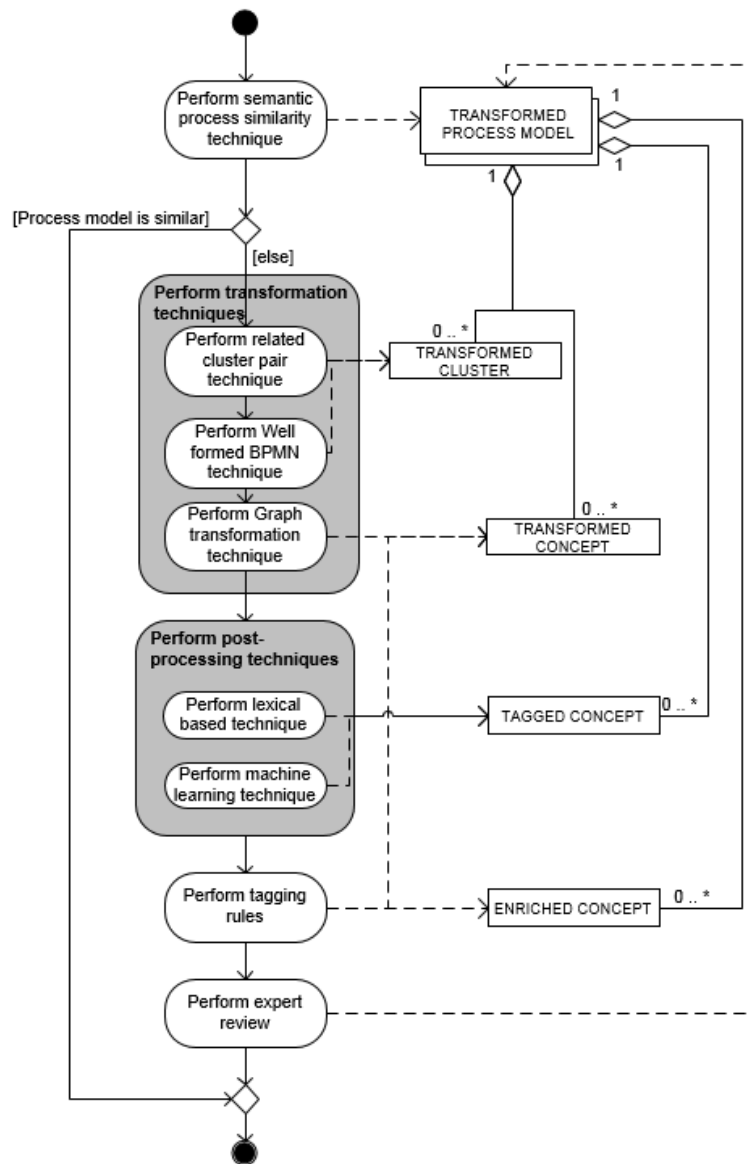


Figure 22: Method for transforming procedural models.

4.3.2 Added activities

The table below (Table 27) contains a description for the added activity. The other activities and concepts remain similar to the initial activities and concepts (Table 25 and Table 26).

Activity name	Sub-activity	Description
Perform expert review		An expert looks at the transformed process models and makes two improvements: removing unnecessary constraints between activities and making activities optional when possible. The changes are stored in the TRANSFORMED PROCESS MODEL concept.

Table 27: Activity table for the final PDD.

5 Discussion

The research performed for this thesis has several challenges and limitations, which are described in this chapter. 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 that is updated as new techniques become available. As such, the techniques used in the method are not considered as exhaustive. At the moment the method consists of six techniques, which can be expanded in future research. For example, Dijkman et al. (2011) expand the similarity measurement with behavioral similarity. In addition to syntactical, semantic and structural similarity, indirect relations between activities are also taken into account. An advanced BPMN querying technique, similar to an internet search engine, has been developed by Awad, Polyvyanyy and Weske (2008). Furthermore, process mining currently results in Petri Net models (van der Aalst, 2011). Research can be done on how to adjust process mining so that it results in a declarative process model. Graph mining algorithms (Yan & Han, 2002) and graph edit distance (Gao, Xiao, Tao, & Li, 2009) might be applicable for retrieving and transforming business processes. 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 that is able to transform multiple source formalisms to multiple target formalisms with a selection custom of techniques (Figure 23).

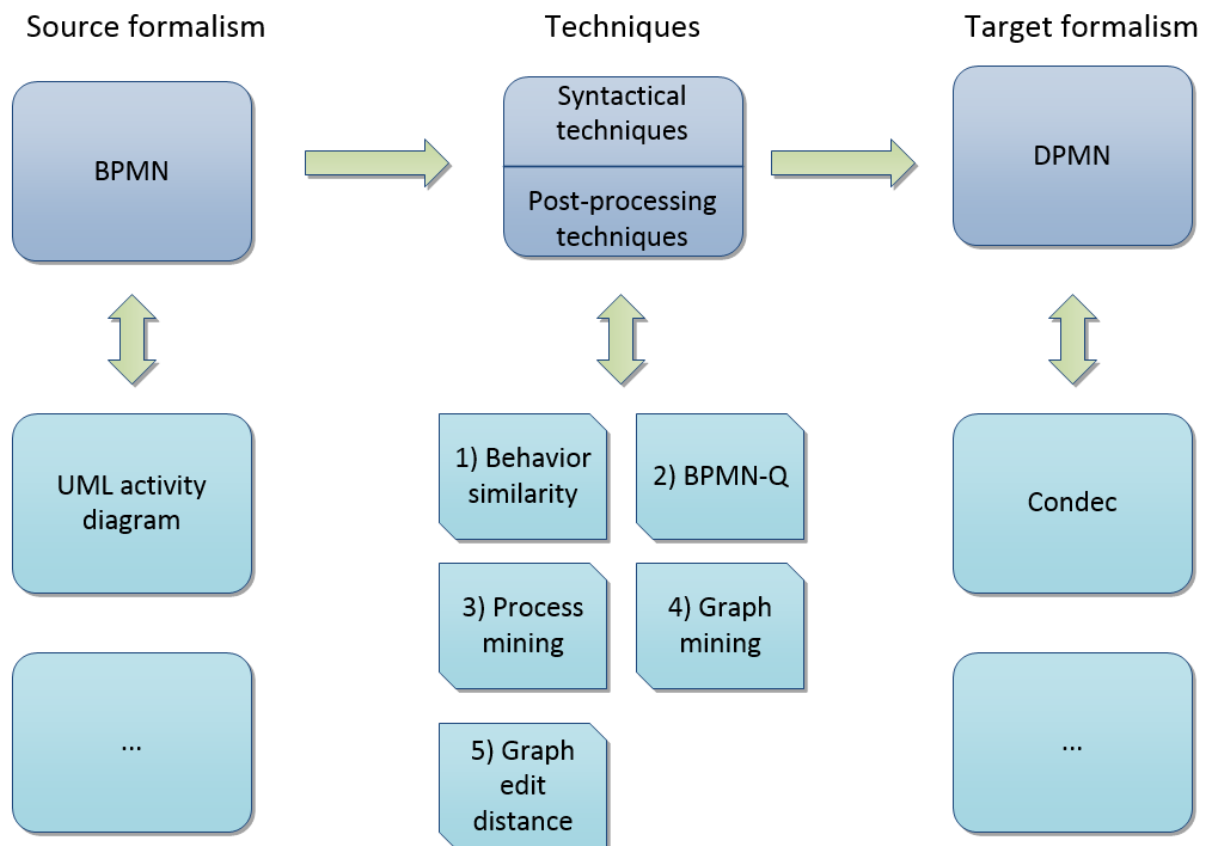


Figure 23: A framework for business process model transformations.

While more techniques can be added to the method, several of the techniques used during this research can be improved. It was not possible to determine a similarity threshold for the semantic process similarity technique, since no queries were formulated for this technique. Further research

could be done to determine from which similarity value a process is considered similar. The lexical analysis of activity labels was partly based on the sample set, which causes uncertainty about the generalizability of the technique. While the lexical analysis technique performed satisfactory during the methodical transformation on the second sample set, more research should be done on the generalizability of the verb classifications. Some of the clusters in the Well-Formed BPMN technique did not occur in the sample sets. In order to make sure these clusters transform properly in DPMN, further experiments could be performed. The classification of activity labels using the Naïve Bayes algorithm during the methodical transformation resulted in many false positives. This could be caused by a limited amount of learning data (517 activity labels). More research should be performed to increase the precision of the Naïve Bayes classifier technique in order to get better results. Lastly, the techniques can be expanded to be compatible with more concepts of the source and target formalisms. For example, pools and lanes were omitted from the transformation due to time constraints. Also, some of the more advanced functions in DPMN are coded in behavior profiles. These were not included in the techniques as well.

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. To make sure these techniques are suitable for a methodical transformation, experiments should be performed where domain specific queries for these techniques are used. 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 further experimenting with the method, researchers should focus on improving the priority of the tags and further improving the enablement fit of the transformation. Lastly, the process models originated from the finance and health domains. It is unclear if the method is also able to transform models from another domain.

Despite these limitations, executing the method results in transformed process models. The transformed models received an average enablement fit of 5.6. This rating means that the DPMN modeling experts consider the transformed models to utilize the available declarative advantages sufficiently. Due to time limitations, the models were only rated by three DPMN modeling experts. To get a better representation of the enablement fit, more DPMN modeling experts should be interviewed. The results of these interviews can be used to get better insight in what parts of the method or tags can be improved.

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 the transformation. This application can be used to automatically transform machine readable BPMN models into DPMN models.

6 Conclusion

The goal of this research was to provide a methodical way of transforming procedural business process models into declarative business process models. To realize this goal, the following research question was formulated:

“How can formal underlying business constraints be extracted from procedural business process models?”

To answer this research question, a start was made by searching for current literature, which was used as a foundation for this research. Based on the current literature, BPMN and DPMN were chosen as source and target formalisms respectively. After that, six techniques were selected and adjusted for transforming or tagging BPMN process models to DPMN models. After performing the techniques on the sample set containing 103 BPMN process models, the techniques were evaluated on their coverage fit. At this point, none of the techniques were discarded, since each technique performed sufficiently and has a unique function.

The adjusted techniques were used to create a set of rules, which were used to create a method. The method was designed to distinguish between transformation and interpretation. Then, the method was used to transform 10 BPMN process models. An evaluation on the performance of the method was used to improve the method. Then, the method was used to transform the 10 BPMN models again. Three DPMN modeling experts were asked to rate how well the transformed process models utilize the declarative advantages of the DPMN formalism. The transformed process models received an average of 5.6 on a scale from 1 to 10. Feedback given by the experts was mostly regarding unnecessary constraints between activities, missing goal orientation and irrelevant activities. An additional activity was added to the method to deal with the first two of these recurring problems.

As such, this research resulted in a method that is able to transform BPMN process models to DPMN models. The method is able to transform most of the BPMN models, except process models with loops. Furthermore, the resulting process models are considered sufficiently utilizing declarative advantages by DPMN modeling experts.

An expansion of the method was also considered. When regarding the method as a framework with interchangeable components it becomes clear that many source formalisms, techniques and target formalisms can be added to the framework. For example, further research should investigate the applicability of graph mining, behavior similarity and other techniques for inclusion in the framework.

In conclusion, the method presented in this research provides the ways and means to transform BPMN process models to DPMN models and has the potential to be more widely applicable by adding components in the future.

7 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.
- Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression trees*. Belmont: Wadsworth.
- Brinkkemper, S. (1996). Method engineering: engineering of information methods and tools. *Information and Software Technology*, 38, 275–280.
- Clarke, E. M., Grumberg, O., & Peled, D. A. (1999). Model Checking. (S. Ramesh & G. Sivakumar, Eds.) *Journal of the American Statistical Association*, 96(5), 314.
- Davenport, T. H. (1993). *Process innovation: reengineering work through information technology*. Harvard Business School Press.
- Davenport, T. H., & Stoddard, D. B. (1994). Reengineering : Business Change of Mythic Proportions ? *MIS Quarterly*, 18(2), 121–127.
- Decker, G., Dijkman, R., Dumas, M., & García-Bañuelos, L. (2008). *Transforming BPMN diagrams into YAWL nets. Business Process Management* (pp. 386–389). Springer Berlin Heidelberg.
- 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., & Ehrig, K. (2006). Overview of Formal Concepts for Model Transformations Based on Typed Attributed Graph Transformation. *Electronic Notes in Theoretical Computer Science*, 152, 3–22.
- 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 Conceptual modelling* (pp. 71–80).
- Fix, E., & Hodges, J. L. (1951). *Discriminatory analysis, nonparametric discrimination*. Randolph Field, Texas.

- Fowler, M., & Beck, K. (1999). *Refactoring: improving the design of existing code*. Addison-Wesley Professional.
- Gao, X., Xiao, B., Tao, D., & Li, X. (2009). A survey of graph edit distance. *Pattern Analysis and Applications*, 13(1), 113–129.
- Giner, P., Torres, V., & Pelechano, V. (2007). Bridging the Gap between BPMN and WS-BPEL. M2M Transformations in Practice. In *Proceedings of the 3rd International Workshop on Model-Driven Web Engineering*.
- 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. (2006). Designing compliant business processes with obligations and permissions. In *Business Process Management Workshops* (pp. 5–14).
- 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).
- Hall, M. A., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1), 10–18.
- Hammer, M. (1990). Reengineering Work: Don't Automate, Obliterate. *Harvard Business Review*, 68(4), 104–112.
- Hammer, M., & Champy, J. (1993). *Reengineering the Corporation: A Manifesto for Business Revolution*. (M. Hammer, Ed.) *Business Horizons* (Vol. 36, pp. 90–91). Harper Business.
- Harrington, H. J., & Harrington, J. S. (1995). *Total improvement management: the next generation in performance improvement*. McGraw-Hill.
- Hevner, A. R., Ram, S., March, S. T., & Park, J. (2004). Design Science in Information Systems Research. *MIS quarterly*, 28(1), 75–105.
- Juristo, N., & Moreno, A. M. (2010). *Basics of software engineering experimentation*. Springer Publishing Company, Inc.
- Kammer, P., Bolcer, G., Taylor, R., Hitomi, A., & Bergman, M. (2000). Techniques for supporting dynamic and adaptive workflow. *Computer Supported Cooperative Work*, 9(3), 1–19.
- Kardasis, P., & Loucopoulos, P. (2004). Expressing and organising business rules. *Information and Software Technology*, 46(11), 701–718.
- Kim, K., Choi, I., & Park, C. (2011). A rule-based approach to proactive exception handling in business processes. *Expert Systems with Applications*, 38(1), 394–409.
- Ko, R. K. L., Lee, S. S. G., & Lee, E. W. (2009). Business process management (BPM) standards : a survey. *Business Process Management Journal*, 15(5), 744–791.

- Kovacic, A. (2004). Business renovation: business rules (still) the missing link. *Business Process Management Journal*, 10(2), 158–170.
- Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet physics doklady*, 10, 707.
- Lu, R., Sadiq, S., & Governatori, G. (2009). On managing business processes variants. *Data & Knowledge Engineering*, 68(7), 642–664.
- 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.
- Miller, G. A. (1995). Wordnet: A lexical Database for English. *Communications of the ACM*, 38(11), 39–41.
- Mutschler, B., Weber, B., & Reichert, M. (2008). Workflow Management versus Case Handling : Results from a Controlled Software Experiment.
- 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.
- Okoli, C., & Schabram, K. (2010). A Guide to Conducting a Systematic Literature Review of Information Systems Research. *Sprouts: Working papers on Information Systems*, 10(26).
- OMG. (2011). *Business Process Model and Notation (BPMN). Version 2.0*. Retrieved from <http://www.omg.org/spec/BPMN/2.0/>
- Ouyang, C., Dumas, M., Ter Hofstede, A., & van der Aalst, W. M. P. (2006). From BPMN Process Models to BPEL Web Services. *2006 IEEE International Conference on Web Services (ICWS'06)*, 285–292.
- Peffer, 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.
- Pichler, P., Weber, B., Zugal, S., Pinggera, J., & Reijers, H. A. (2012). Imperative versus Declarative Process Modeling Languages : An Empirical Investigation. In *Business Process Management Workshops* (pp. 383–394). Springer Berlin Heidelberg.
- Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann.

- Recker, J. (2008). BPMN Modeling – Who , Where , How and Why. *BPTrends*, (March), 1–8.
- 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*.
- Reichert, M., & Dadam, P. (1998). ADEPT flex—supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems*, 10(2), 93–129.
- Russell, N., Hofstede, A. Ter, & Mulyar, N. (2006). Workflow controlflow patterns: A revised view.
- Schonenberg, M. H., Mans, R. S., Russel, N. C., Mulyar, N. A., & van der Aalst, W. M. P. (2008). Towards a Taxonomy for Process Flexibility. In *Caise* (pp. 81–84).
- Sinur, J. (2012). Business Rules Get No Respect. Retrieved April 11, 2013, from http://blogs.gartner.com/jim_sinur/2012/12/11/business-rules-get-no-respect/
- Smirnov, S., Reijers, H. a., Weske, M., & Nugteren, T. (2012). Business process model abstraction: a definition, catalog, and survey. *Distributed and Parallel Databases*, 30(1), 63–99.
- Stoddard, D. B., & Jarvenpaa, S. L. (1995). Business Process Redesign: Tactics for Managing Radical Change. *Journal of Management Information Systems*, 12(1), 81–107.
- 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). Hershey, Idea Group Publishing.
- Van der Aalst, W. M. P. (1998a). the Application of Petri Nets To Workflow Management. *Journal of Circuits, Systems and Computers*, 08(01), 21–66.
- Van der Aalst, W. M. P. (1998b). Three good reasons for using a Petri-net-based workflow management system. In T. Wakayama (Ed.), *Information and Process Integration in Enterprises: Rethinking documents* (Vol. 08, pp. 161–182). Norwell: Kluwer Academic Publishers.
- Van der Aalst, W. M. P. (2011). *Process Mining*. Springer-Verlag Berlin Heidelberg.
- Van der Aalst, W. M. P., & Jablonski, S. (2000). Dealing with workflow change : identification of issues and solutions. *Computer Systems Science & Engineering*, 15(5), 267–276.
- Van der Aalst, W. M. P., Pesic, M., & Schonenberg, H. (2009). Declarative workflows: Balancing between flexibility and support. *Computer Science - Research and Development*, 23(2), 99–113.

- 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., & Rensen, G. (2013). *Towards Webscale Business Processes*.
- Varró, D., Varró, G., & Pataricza, A. (2002). Designing the automatic transformation of visual languages. *Science of Computer Programming*, 44(2), 205–227.
- Verschuren, P. J. M., & Doorewaard, H. (2007). *Het ontwerpen van een onderzoek*. Lemma.
- Weber, B., Mutschler, B., & Reichert, M. (2010). Investigating the effort of using business process management technology: Results from a controlled experiment. *Science of Computer Programming*, 75(5), 292–310.
- Weber, B., Reichert, M., Mendling, J., & Reijers, H. a. (2011). Refactoring large process model repositories. *Computers in Industry*, 62(5), 467–486.
- Webster, J., & Watson, R. T. (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly*, 26(2), 13–24.
- Weske, M. (2001). Formal foundation and conceptual design of dynamic adaptations in a workflow management system. In *System Sciences, 2001. Proceedings of the 34th annual Hawaii International Conference*. Los Alamitos: IEEE Comput. Soc.
- Weske, M. (2007). *Business Process Management: Concepts, Languages, Architectures*. (Springer Berlin Heidelberg New York, Ed.) *Business Process Management* (Vol. 54, p. 368). Springer.
- Weske, M., van der Aalst, W. M. P., Verbeek, H. M. W., & Aalst, W. M. P. (2004). Advances in business Process Management. *Data & Knowledge Engineering*, 50(1), 1–8.
- White, S. (2004). *Business Process Modeling Notation (BPMN). Version 1.0*. BPMI.org. Retrieved from www.bpmi.org
- 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.
- Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., ... Steinberg, D. (2007). *Top 10 algorithms in data mining*. *Knowledge and Information Systems* (Vol. 14, pp. 1–37).
- Yan, X., & Han, J. (2002). gSpan: Graph-Based Substructure Pattern Mining. In *IEEE International conference on Data Mining* (pp. 721–724). IEEE.

- Zeist, R. H. J., & Hendriks, P. R. H. (1996). Specifying software quality with the extended ISO model. *Software Quality Journal*, 5(4), 273–284.
- Zoet, M., Versendaal, J., Ravesteyn, P., & Welke, R. (2011). Alignment of Business Process Management and Business Rules Management. In *European Conference on Information Systems (ECIS)*.
- Zugal, S., Pinggera, J., & Weber, B. (2011). The Impact of Testcases on the Maintainability of Declarative Process Models. In *Enterprise, Business-Process and Information Sysyems Modeling* (pp. 163–177). Springer Berlin Heidelberg.
- 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.

Appendix A: Systematic literature review protocol

This research protocol is based on the eight-step guide to conducting a systematic literature review (Okoli & Schabram, 2010) In order to gather and assess literature sources in a uniform and reproducible fashion, the protocol is followed concisely.

The first step in the systematic literature review is a broad search for literature. In order to get an adequate level of knowledge on the subject of this thesis, three subjects with multiple keywords were chosen for search engine input. The subjects are BPM, flow-based formalisms and constraint-based formalisms and correspond several of the foundations depicted in the research model. Google Scholar and Omega from the University of Utrecht were used as search engines. For every search request, the first 30 results were considered. The broad search resulted in 56 papers, which were stored and managed in Mendeley. This includes papers received from or recommended by the thesis supervisors. The second step in the systematic literature review is filtering out papers based on global criteria. This includes the topic of the article, publication date and findings in the article. Then the articles are filtered again on if the medium (such as journals or conference proceedings) peer-reviews the articles they publish. Then the important information in the articles is annotated and lastly, the information is synchronized into a literature review. During the information annotation step, interesting references within that paper were also looked up and considered for the systematic literature review.

1. Searching for the literature:

The first step focuses on finding as much relevant articles as possible; the following criteria are used for searching:

Search Sources	
Source:	URL:
Google Scholar	http://scholar.google.com/
Omega	http://Omega.library.uu.nl

Search Keywords*	
Topic:	Keywords/Synonyms:
Business Process Management**	BPM, Literature review BPM, Review BPM, state of the art BPM, state BPM, Literature BPM,
flow-based formalisms	Goal-driven BPM, Goal-oriented BPM, van der Aalst
Constraint-based formalisms	Constraint based business process, Constraint based process formalism, case-handling business process,
Other	Types of Business process notations, Business Process formalisms

*For every search, also use UK-English alternative

** BPM keywords were used abbreviated and fully written out during the literature search.

2. Practical screen:

The second step focuses on filtering the articles found in step 1 on several global criteria. Often it will suffice to only read the abstract and conclusion of the article in order to assess the potential value for this the research.

Inclusion Criteria	
Criterion:	Value:
Article is on topic	
The article contains relevant findings	
The article has been publishing fairly recently	1990 - 2013

3. Quality appraisal:

In The third step go in to more detail and try to leave out articles that do not meet the needed quality standards for this research. The main quality factors are:

Quality Factors	
Factor:	Example:
Research method	Literature study, survey, case study, experiment
Publishing media	Journal, conference proceeding, book
Generalizability of findings	Can the findings easily be generalized and applied to different situation?
Profoundness	Contains a framework or explicit method
Validity	Are the results valid?

4. Data extraction:

The fourth step focuses on extracting applicable information from the articles that remained of interest after steps 2 and 3. The articles are read carefully and important parts, relevant to our research objectives are annotated

5. Synthesis of studies

Finally all relevant data extracted in step 4 is aggregated. Articles are grouped under the (sub)topics of this thesis.

Appendix B: Workflow patterns

As a further demonstration of the differences between flow-based and constraint-based formalisms, modeled patterns recurring in flow-based formalism have been modeled in a constraint-based formalism. The control-flow patterns are adapted from Russell, Hofstede, & Mulyar (2006) and are modeled in this section according to the BPMN and DPMN formalisms. However, there is a mismatch between these workflow patterns and constraint-based formalisms, such as the DPMN formalism. This will be demonstrated by attempting to model them and explain what is wrong with the declarative models.

The first workflow pattern, shown in Table 28 below, depicts a sequence pattern. In BPMN, a sequence pattern is modeled by two consequent activities with a sequence flow between them. In DPMN, there is a case where activities are performed. In this example, Activity 2 requires Activity 1 to be completed, thus creating a sequence of activities. Alternative methods of modeling sequence DPMN exist. For example, when a pre-condition for Activity 2 is the requirement of a post-condition of Activity 1 the order of activities is modeled implicitly.

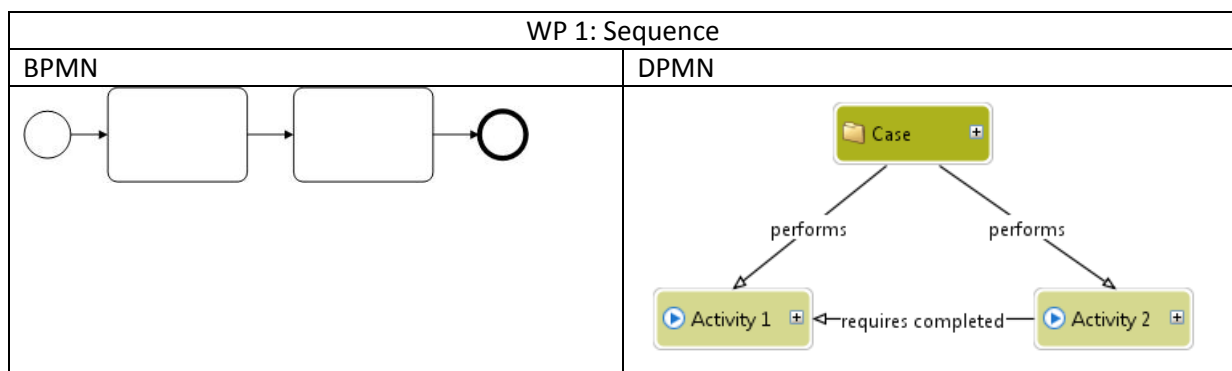


Table 28: Workflow pattern 1: Sequence.

In Table 29 the parallel split pattern is modeled. In BPMN, a parallel split is modeled by using the AND gateway with two outgoing activities. In DPMN, the notion of splitting a flow does not exist. However, the effect of performing two activities parallel to each other can be achieved by modeling a case that performs two activities and no explicit and implicit conditions are modeled.

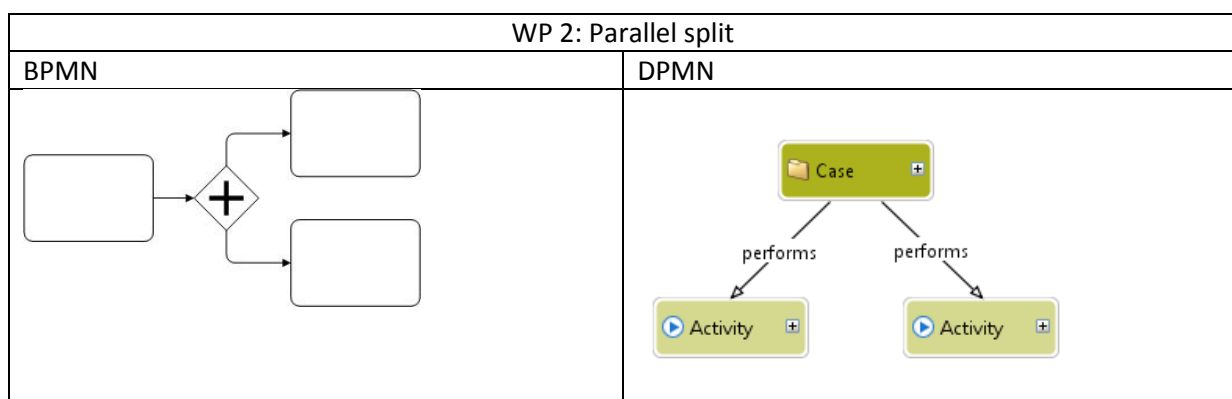


Table 29: Workflow pattern 2: Parallel split.

The third workflow pattern is the synchronization pattern and can be viewed in Table 30. The synchronization pattern occurs when two flow branches merge into one branch and the flow continues when all merging branches have been completed. In BPMN the synchronization pattern is modeled by using the AND gateway to join the branches. In DPMN this is modeled by adding the

“requires completed” pre-condition from the merged activity to the activities in the merging branches.

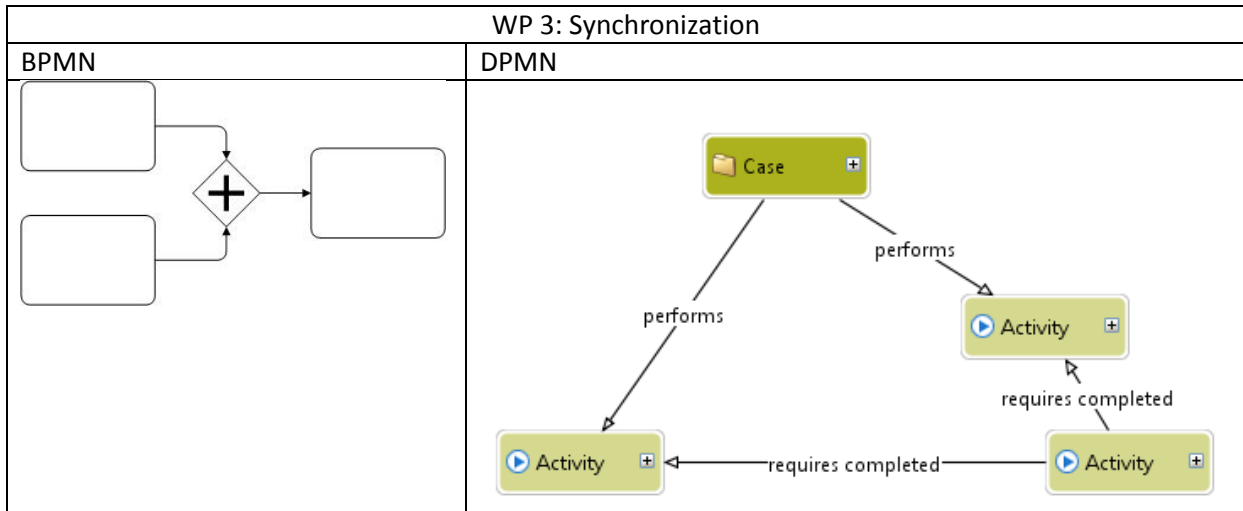


Table 30: Workflow pattern 2: Synchronization.

The fourth workflow pattern is an exclusive choice and can be viewed in Table 31. In BPMN, an exclusive choice is modeled with an XOR gateway. The gateway has a default sequence flow and a sequence flow with a condition (For example: $X > Y$). In DPMN, two activities are performed again. Both activities have a constraint on the “performs” relation. The value of x is determined by the “determine x ” decision concept.

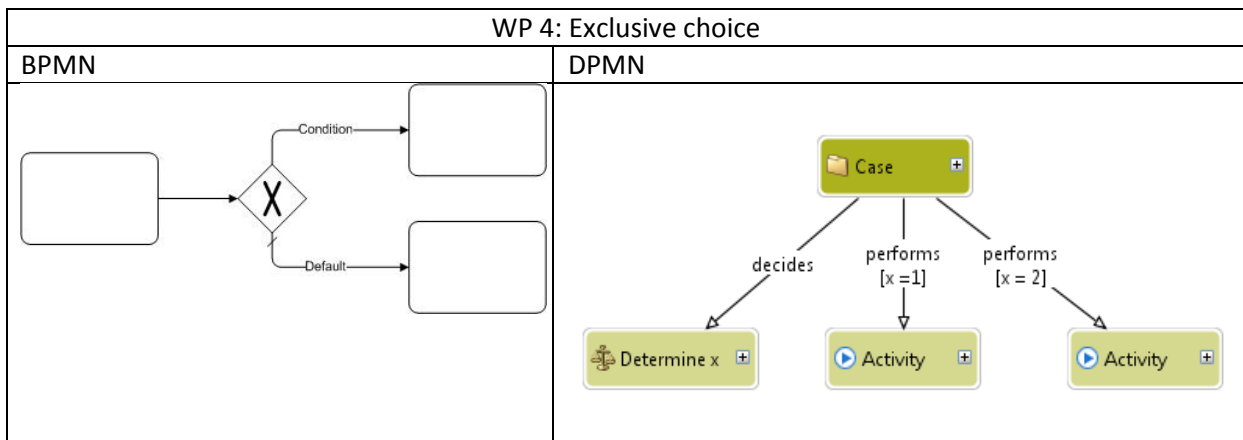
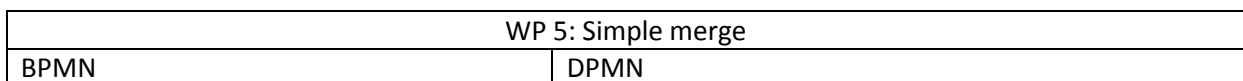


Table 31: Workflow pattern 3: Exclusive choice.

Workflow pattern 5 is depicted in Table 32 and depicts a simple merge. In a simple merge, each incoming branch activates the next incoming branch, so it is not needed to complete each incoming branch before continuing. In BPMN a simple merge is modeled with an XOR gateway. In DPMN this is modeled by using the “implied by completed” constraint on the exclusive activities. When an exclusive activity is completed, the next activity can be performed.



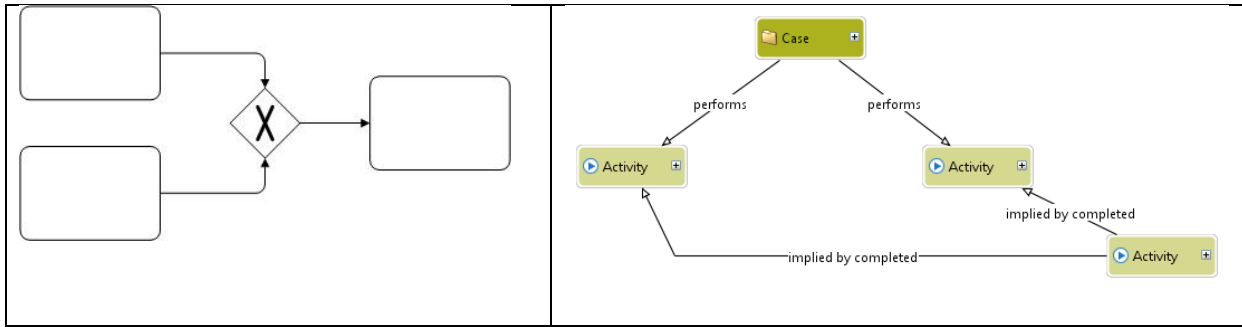


Table 32: Workflow pattern 5: Simple merge.

Workflow pattern 6 is shown in Table 33. The multiple choice pattern is used when multiple branches are accessible from one single branch. In BPMN, the multiple choice pattern is depicted by using the OR gateway. The branches are selected based on conditions placed on the sequence flows. In DPMN, this is achieved by adding a condition to the “performs” relationship. When a condition is met in a “performs” relationship, the corresponding activity has to be performed. However, when the condition is not met, the activity can still be performed. This is determined by the user at run-time and offers flexibility at execution time.

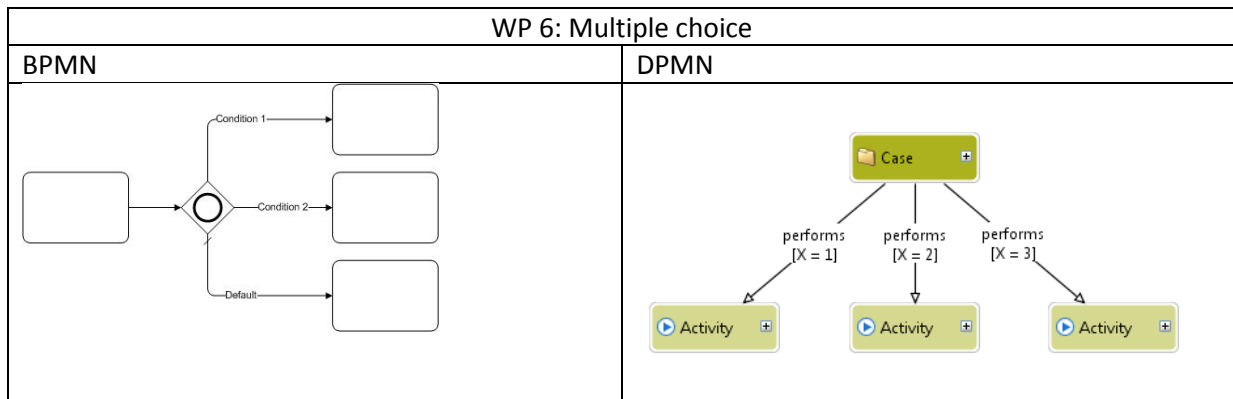


Table 33: workflow pattern 6: Multiple choice.

Table 34 shows the synchronizing merge pattern. In this pattern, the branches which were activated in the preceding multiple choice pattern are merged. The process continues when all activated branches are completed. In BPMN, the synchronizing merge pattern is modeled by using the OR gateway to merge the incoming branches. In DPMN this is more complicated, because the “requires available” and “implied by available” relations do not fully support this pattern. To solve this, an activity is modeled, which has to create a document concept when the condition is met. The creation of a document is often depicted as an activity in a control-flow, but is modeled separately in DPMN. When the creation of applicable documents is done, the activity is completed and the next activity can be performed.

WP 7: Synchronising merge	
BPMN	DPMN

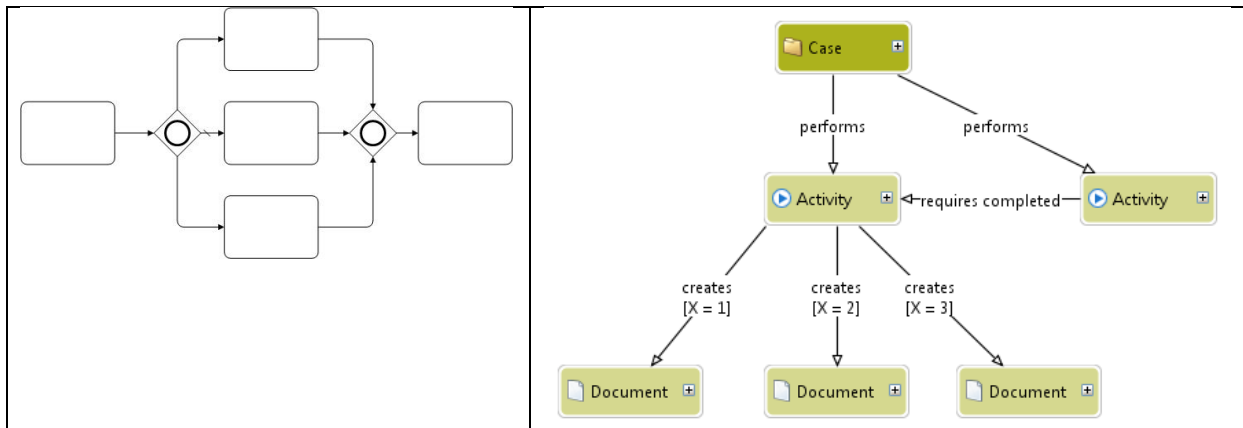


Table 34: Workflow pattern 7: Synchronizing merge.

The multiple merge pattern is depicted in Table 35. The difference with the synchronizing merge is that any incoming branch causes the flow to continue to the next activity. In BPMN, the multiple merge pattern is modeled with an XOR gateway. In DPMN the “*implied by available*” relation is used on any of the created documents. When any of the documents is created, the next activity can be performed. However, this activity can only be performed once, regardless of the amount of incoming branches.

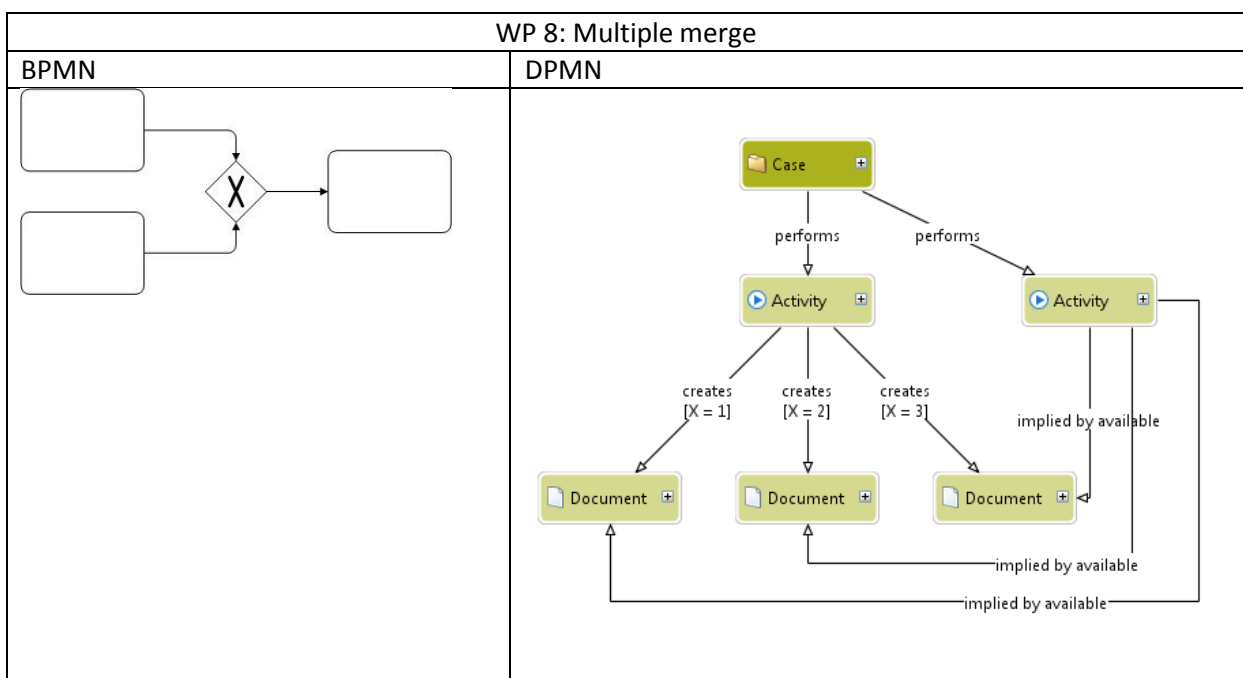


Table 35: Workflow pattern 8: Multiple merge.

The discriminator pattern is shown in Table 36. The discriminator pattern is used when multiple branches are converged into a single branch and only the first incoming branch is used to continue with the flow. The discriminator pattern is preceded by a parallel split. In BPMN a multiple instances type activity is used to depict the multiple incoming branches. When the first instance is completed the flow continues with the next activity. In DPMN, the discriminator pattern is modeled with an “*implied by available*” relation. When any of these documents are available the activity can be performed.

WP 9: Discriminator

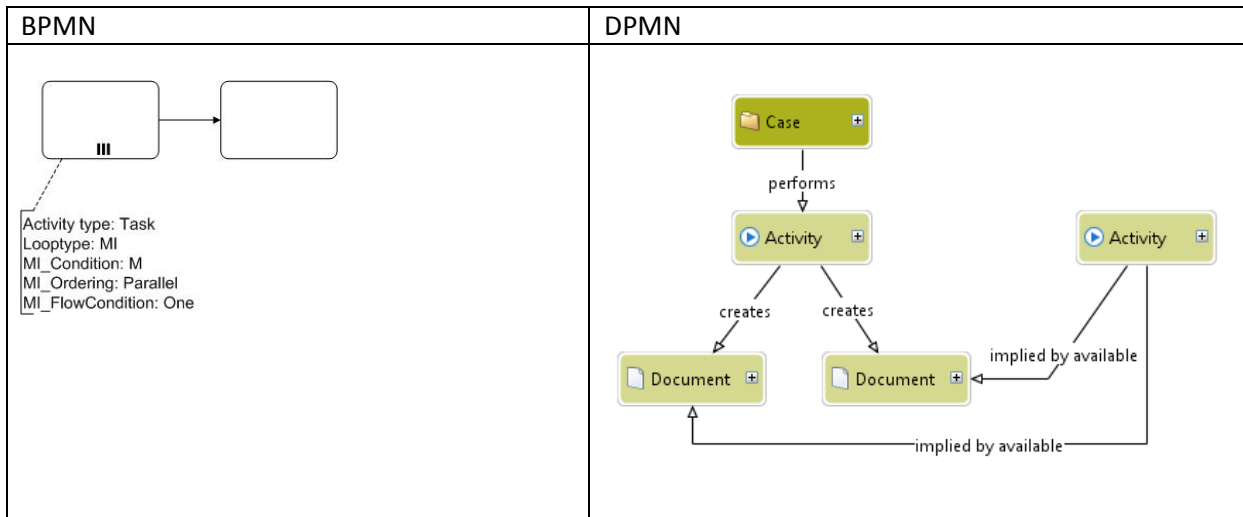


Table 36: Workflow pattern 9: Discriminator.

Table 37 depicts the arbitrary cycle pattern. An arbitrary cycle represents the looping or iterating through a set of activities. It is represented with an XOR splits and joins in BPMN. It is currently not possible to model this pattern in DPMN.

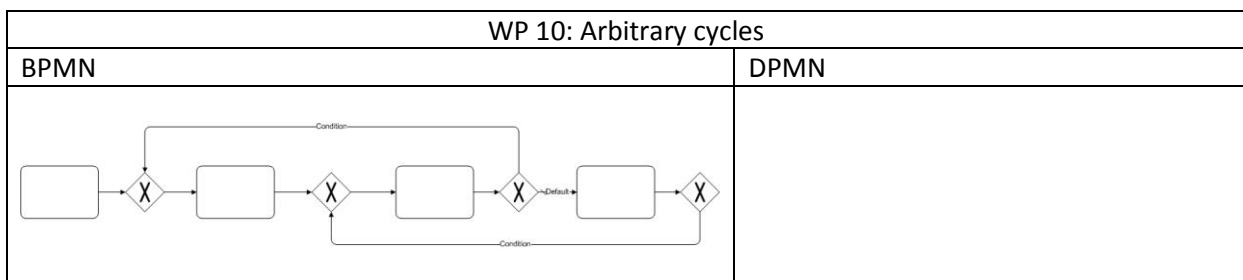


Table 37: Workflow pattern 10: Arbitrary cycles.

The implicit termination pattern is shown in Table 38. This pattern represents if there is an objective means of determining of the model has finished executing. In BPMN, end nodes can be used to model this pattern. In DPMN this pattern is modeled implicitly. When all activities are performed or the activities are not applicable now and in the future, the process is automatically completed.

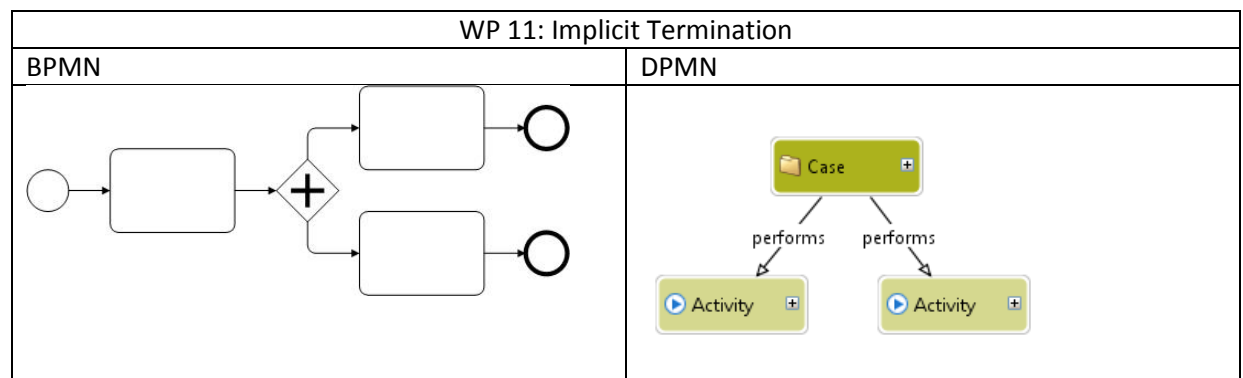


Table 38: Workflow pattern 10: Implicit termination.

Table 39 depicts patterns related to multiple instances. In these types of patterns, the same activity is performed multiple times during an instantiation of the model. This is modeled with a multiple instance activity in BPMN. It is not possible to model these patterns in DPMN. Furthermore, workflow pattern 15 is not supported by both BPMN and DPMN.

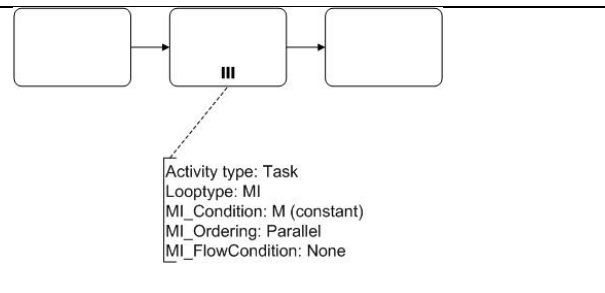
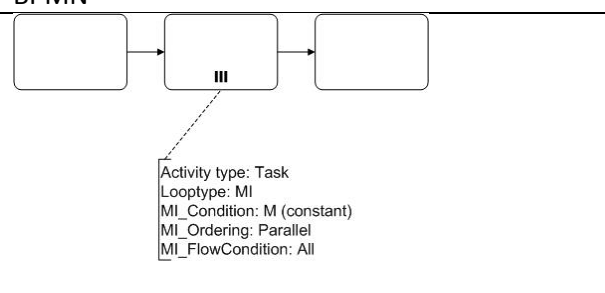
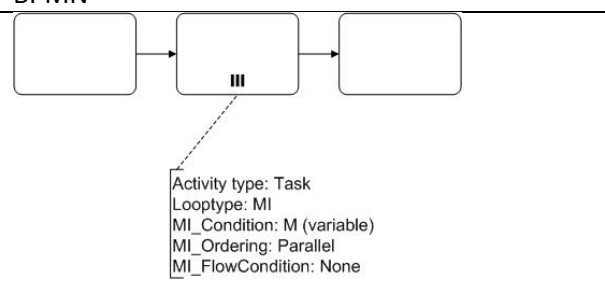
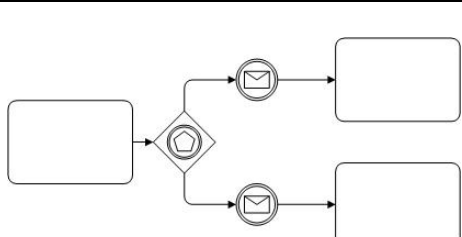
WP 12: Multiple instances without synchronization	
BPMN	DPMN
 <p>Activity type: Task Looptype: MI MI_Condition: M (constant) MI_Ordering: Parallel MI_FlowCondition: None</p>	
WP 13: Multiple instances with a priori design time knowledge	
BPMN	DPMN
 <p>Activity type: Task Looptype: MI MI_Condition: M (constant) MI_Ordering: Parallel MI_FlowCondition: All</p>	
WP 14: Multiple instances with a priori runtime knowledge	
BPMN	DPMN
 <p>Activity type: Task Looptype: MI MI_Condition: M (variable) MI_Ordering: Parallel MI_FlowCondition: None</p>	
WP 15: Multiple instances without a priori runtime knowledge	
BPMN	DPMN

Table 39: Three multiple instances patterns.

The deferred choice pattern is depicted in Table 40. In the deferred choice pattern, a split branch is executed based on an external event. As such, there is not an explicit decision moment, but the branch to be executed is determined by which event occurs first. In BPMN, the deferred choice pattern is modeled with the event gateway and the intermediate message event. In DPMN, an activity can be triggered by a business event. By adding an “*excluded by completed*” relation between activities, exclusivity is realized.

WP 16: Deferred choice	
BPMN	DPMN
	

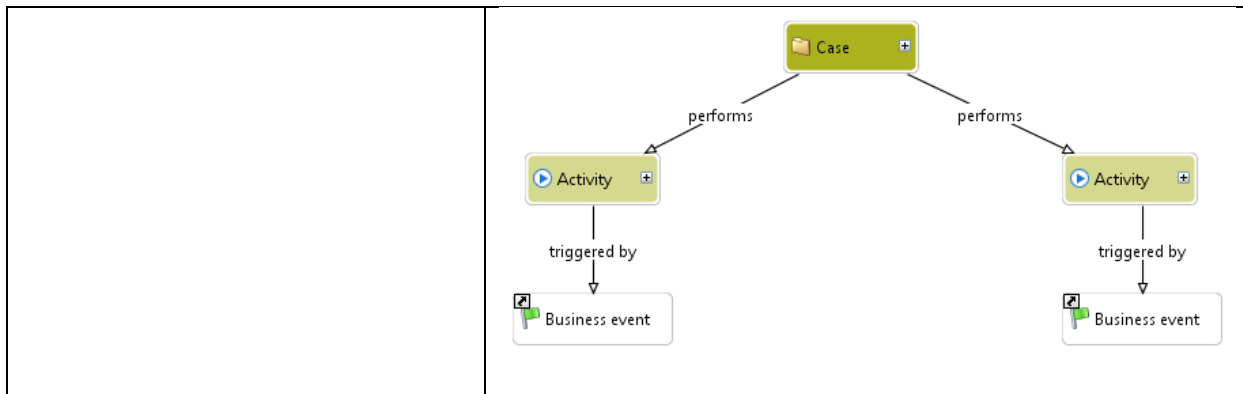


Table 40: Workflow pattern 16: Deferred choice.

Table 41 depicts the interleaved parallel routing pattern. In this pattern, there are multiple tasks that can be performed in any order. The only restriction is that only one task can be performed at the same time. In BPMN this pattern is partially modeled with an ad-hoc task. However, the random execution of sequences of tasks cannot be modeled according to this pattern. DPMN fully supports this pattern, because activities can be performed in any order when no constraints are given.

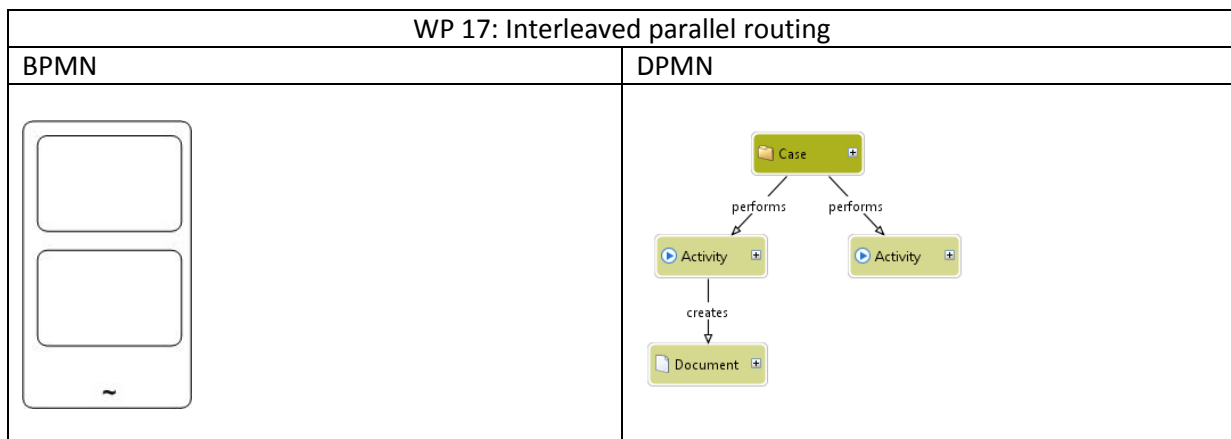


Table 41: Workflow pattern 17: Interleaved parallel routing.

Workflow pattern 18, the milestone pattern, cannot be depicted in either BPMN or DPMN. The next workflow pattern is the cancel activity and is depicted in Table 42. The cancel activity pattern is used when an activity is canceled prior or during the execution of the activity. This pattern is modeled with an error event attached to the corresponding activity. There is now specific way to model this pattern in DPMN, but it is possible to cancel an activity during execution time.

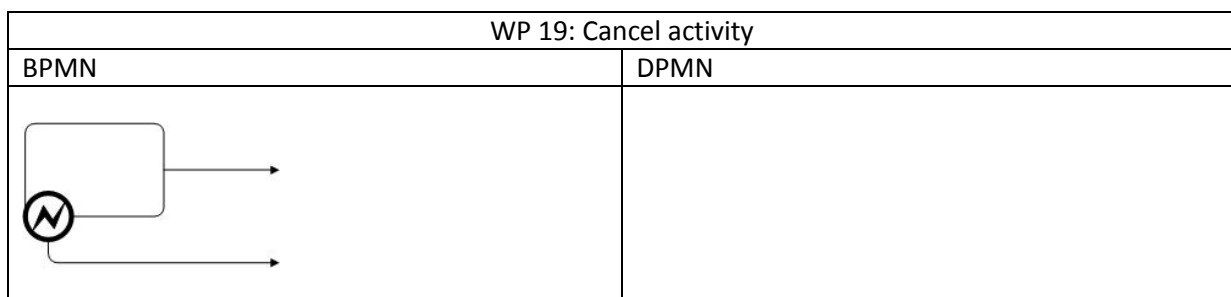


Table 42: Workflow pattern 19: Cancel activity.

The last workflow pattern discussed in this document is the cancel case pattern and can be viewed in Table 43. The pattern is concerned with the cancelation of the entire process. In BPMN this pattern is

modeled by a termination event, either attached to an activity or in the sequence flow. There is no direct way to model this in DPMN. However, a cancellation of the process can be achieved when a time limit is reached. If the process is not completed before the time limit is reached and the limit is not suspended or ended, the case will be canceled.

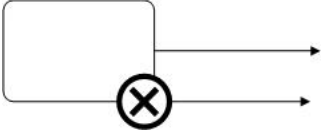
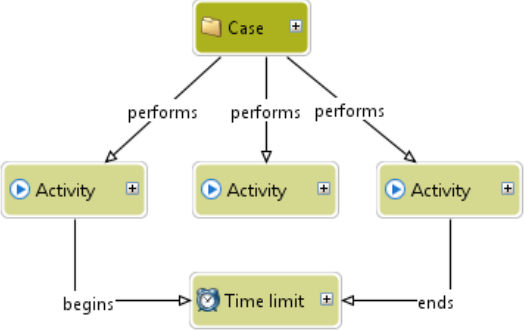
WP 20: Cancel case	
BPMN	DPMN
	

Table 43: Workflow pattern 20: Cancel case.

Appendix C: Code used for calculating similarity values

1.1	Related cluster pair technique	66
1.1.1	Levenshtein distance metric.....	66
1.1.1.1	Levenshtein()	66
1.1.2	Monge Elkan similarity metric (with synonyms)	66
1.1.2.1	Simme()	66
1.1.2.2	Synonym().....	67
1.2	Semantic process similarity technique.....	67
1.2.1	Levenshtein	67
1.2.1.1	Simsyn()	67
1.2.1.2	Levenshtein()	68
1.2.2	Linguistic similarity	68
1.2.2.1	Simling().....	68
1.2.2.2	Rdupes().....	69
1.2.2.3	Sen().....	70
1.2.2.4	Cardinalities()	71
1.2.3	Simcom and Simstr	71
1.2.3.1	Simcom().....	71
1.2.4	Monge Elkan similarity metric.....	72
1.2.4.1	Melkan()	72

1.1 Related cluster pair technique

1.1.1 Levenshtein distance metric

1.1.1.1 *Levenshtein()*

```
Function Levenshtein(ByVal string1 As Range, ByVal string2 As String) As Double

'Adapted from http://stackoverflow.com/questions/4243036/levenshtein-distance-in-excel

'declarations
Dim i As Long, j As Long
Dim comstring1 As String
Dim counter As Long
Dim string1_length As Long
Dim string2_length As Long
Dim distance() As Long
Dim editdistance As Long
Dim maxstring As Long

'loading values
comstring1 = combine(string1)
string1_length = Len(comstring1)
string2_length = Len(string2)
ReDim distance(string1_length, string2_length)

'calculating distance
For i = 0 To string1_length
    distance(i, 0) = i
Next

For j = 0 To string2_length
    distance(0, j) = j
Next

For i = 1 To string1_length
    For j = 1 To string2_length
        If Asc(Mid$(comstring1, i, 1)) = Asc(Mid$(string2, j, 1)) Then
            distance(i, j) = distance(i - 1, j - 1)
        Else
            distance(i, j) = Application.WorksheetFunction.Min _
                (distance(i - 1, j) + 1, _
                 distance(i, j - 1) + 1, _
                 distance(i - 1, j - 1) + 1)
        End If
    Next
Next

'calculating Simlev
Levenshtein = 1 - distance(string1_length, string2_length) / Application.WorksheetFunction.Max(string1_length, string2_length)

End Function
```

1.1.2 Monge Elkan similarity metric (with synonyms)

1.1.2.1 *Simme()*

```
Function simme(ByVal string0 As Range, ByVal string1 As Range, ByVal string2 As String) As Double

'declarations
Dim originals() As Variant
Dim target As String
Dim synonyms As String
Dim counter As Double
Dim mkan As Double

'loading values
target = string2
originals() = string0
counter = 1

'calculate synonym equation value for each word. calls Synonym() function
For Each original In originals()

    synonyms = string1(1, counter).value
    mkan = mkan + Synonym(original, synonyms, target)
    counter = counter + 1
Next

'calculate Monge Elkan equation value
simme = (1 / (counter - 1)) * mkan

End Function
```

1.1.2.2 *Synonym()*

```
Function Synonym(ByVal string0 As String, ByVal string1 As String, ByVal string2 As String) As Double
```

```
'declarations
Dim original() As String
Dim synonyms() As String
Dim row() As String
Dim target() As String
Dim simscore As Double
Dim divider As Double
Dim found As Boolean

'loading values
original() = Split(string0, " ")
synonyms() = Split(string1, " ")
target() = Split(string2, " ")
found = False

'calculate synonym value
For Each orword In original
'For loop for separating original word from its synonyms
'determines if words are equal
  For Each word In target()
    If word = orword And found = False Then
      simscore = simscore + 1
      found = True
      Exit For
    End If
  Next
Next

'determines if target word is synonym
For Each syn In synonyms()

  For Each word In target()
    If word = syn And found = False Then
      simscore = simscore + 0.5
      found = True
      Exit For
    End If
  Next

Next

Synonym = simscore
End Function
```

1.2 Semantic process similarity technique

1.2.1 Levenshtein

1.2.1.1 *Simsyn()*

```
Function simsyn(ByVal l1 As String, ByVal l2 As String) As Double
```

```
'declarations
Dim lev As Double
Dim label1 As String
Dim label2 As String
Dim length1 As String
Dim length2 As String
Dim min As Double

'loading values
label1 = l1
label2 = l2
length1 = Len(label1)
length2 = Len(label2)
min = Application.WorksheetFunction.min(length1, length2)
lev = levenshtein(label1, label2)

similarity = (min - lev) / min
'returning the calculation
simsyn = Application.WorksheetFunction.Max(0, similarity)

End Function
```

1.2.1.2 Levenshtein()

```
Function levenshtein(ByVal string1 As String, ByVal string2 As String) As Long
'declarations
Dim i As Long, j As Long
Dim string1_length As Long
Dim string2_length As Long
Dim distance() As Long
'loading values
string1_length = Len(string1)
string2_length = Len(string2)
ReDim distance(string1_length, string2_length)
'Calculating Levenshtein distance. Adapted from http://stackoverflow.com/questions/4243036/levenshtein-distance-in-excel
For i = 0 To string1_length
    distance(i, 0) = i
Next

For j = 0 To string2_length
    distance(0, j) = j
Next

For i = 1 To string1_length
    For j = 1 To string2_length
        If Asc(Mid$(string1, i, 1)) = Asc(Mid$(string2, j, 1)) Then
            distance(i, j) = distance(i - 1, j - 1)
        Else
            distance(i, j) = Application.WorksheetFunction.min _
                (distance(i - 1, j) + 1, _
                 distance(i, j - 1) + 1, _
                 distance(i - 1, j - 1) + 1)
        End If
    Next
Next

'returning levenshtein distance
levenshtein = distance(string1_length, string2_length)

End Function
```

1.2.2 Linguistic similarity

1.2.2.1 Simling()

```
Function simling(ByVal string3 As String, ByVal string4 As String) As Double
'declarations
Dim fs As Double
Dim sim As Double
Dim string1length As Double
Dim string2length As Double
Dim arr() As String
Dim senses() As String

'removes words if they are in both labels and when the target string (string4) consists of multiple words
arr() = rdupes(string3, string4)

'retrieve all senses for that word. uses sen() function
If arr(0) <> Empty And arr(1) <> Not Empty Then

    senses() = sen(arr(0), arr(1))
    'intersection between senses?
    If senses(0) <> Empty And senses(1) <> Empty Then

        If senses(0) = senses(1) Then
            'if the words are similar a value of 1 is returned
            simling = 1
        Else
            'if the words are not similar, the simling value is calculated
            'Cardinalities() returns 1 if there is the words share a sense, otherwise 0
            fs = cardinalities(senses(0), senses(1))
            string1length = UBound(Split(senses(0), " ")) + 1
            string2length = UBound(Split(senses(1), " ")) + 1
            sim = fs / Application.WorksheetFunction.Max(string1length, string2length)
            sim = Application.WorksheetFunction.Round(sim, 6)
            simling = sim
        End If
    Else
        simling = 0
    End If
Else
    simling = 0
End If

End Function
```

1.2.2.2 Rdupes()

```
Function rdupes(ByVal string1 As String, ByVal string2 As String) As String()
'declarations
Dim astring1() As String
Dim astring2() As String
Dim col As New Collection
Dim col2 As New Collection
Dim add As Double
Dim counter As Double
Dim placeholder As String
Dim placeholder2 As String
Dim placeholder3(0 To 1) As String
Dim dummy() As String
Dim dummycounter As Double

'loading values
astring1 = Split(string1, " ")
astring2 = Split(string2, " ")
dummycounter = 0

'counting how many words are in string2
For Each dum In astring2()
    dummycounter = dummycounter + 1
Next

add = 1
counter = 1

'removing words occuring in both labels if string2 has more than 1 words
If dummycounter > 1 Then

    For Each w1 In astring1
        col.add (w1)
    Next

    For Each w2 In astring2
        For Each c1 In col
            If w2 = c1 Then
                add = 0
                col.Remove (counter)
            End If
            counter = counter + 1
        Next
        If add = 1 Then
            col2.add (w2)
        Else
            add = 1
        End If
        counter = 1
    Next

    counter = 0
    For Each c2 In col
        If counter = 0 Then
            placeholder = c2
            counter = 1
        Else
            placeholder = placeholder + " " + c2
        End If
    Next

    counter = 0
    For Each c3 In col2
        If counter = 0 Then
            placeholder2 = c3
            counter = 1
        Else
            placeholder2 = placeholder2 + " " + c3
        End If
    Next

    placeholder3(LBound(placeholder3)) = placeholder
    placeholder3(UBound(placeholder3)) = placeholder2

    rdupes = placeholder3()
Else
    placeholder3(LBound(placeholder3)) = string1
    placeholder3(UBound(placeholder3)) = string2
    rdupes = placeholder3()
End If
End Function
```

1.2.2.3 Sen()

```
Function sen(ByVal words1 As String, ByVal words2 As String) As String()
'declarations
Dim w1() As String
Dim w2() As String
Dim s1 As String
Dim s2 As String
Dim s1new As Double
Dim s2new As Double
Dim sense() As Variant
Dim counter As Double
Dim senses(0 To 1) As String
'loading values
s1new = 0
s2new = 0
sense() = Range("P69:Q321")
w1 = Split(words1, " ")
w2 = Split(words2, " ")
'returning senses for label 1 (w1) and label 2 (2)
For i = 1 To UBound(sense())
    For Each w11 In w1
        If w11 = sense(i, 1) Then

            If s1new = 0 Then
                s1 = sense(i, 2)
                s1new = 1
            Else
                s1 = s1 + " " + sense(i, 2)
            End If
        End If
    Next
    For Each w22 In w2
        If w22 = sense(i, 1) Then

            If s2new = 0 Then
                s2 = sense(i, 2)
                s2new = 1
            Else
                s2 = s2 + " " + sense(i, 2)
            End If
        End If
    Next

    senses(i) = s1 & " " & s2
Next

sen = senses()

End Function
```

1.2.2.4 Cardinalities()

```
Function cardinalities(ByVal string0 As String, ByVal string1 As String) As Double

Dim synset1() As String
Dim synset2() As String
Dim intersection() As String
Dim counter As Double

synset1() = Split(string0, " ")
synset2() = Split(string1, " ")
counter = 0
For Each syn In synset1

    For Each syn2 In synset2
        If syn = syn2 Then

            'ReDim Preserve intersection(0 To UBound(intersection) + 1) As String
            ReDim Preserve intersection(0 To counter)
            intersection(counter) = syn
            counter = counter + 1
            End If
        Next
    Next

If counter >= 1 Then
    cardinalities = 1
Else
    cardinalities = 0
End If

End Function
```

1.2.3 Simcom and Simstr

1.2.3.1 Simcom()

```
Function simcom(ByVal l1 As String, ByVal l2 As String, ByVal l3 As String, ByVal l4 As String) As Double

'declarations
Dim label1 As String
Dim label2 As String
Dim label3 As String
Dim label4 As String
Dim syn As Double
Dim ling As Double
Dim str As Double
'label 1 = query label
label1 = l1
'label 2 = target label
label2 = l2
'label 3 = successor querylabel
label3 = l3
'label 4 = successor target label
label4 = l4

'loading values
syn = simsyn(label1, label2)
ling = simling(label1, label2)
'calculating Simstr
If label3 <> Empty And label4 <> Empty Then
    str = Application.WorksheetFunction.Max(simsyn(label3, label4), simling(label3, label4))
    simcom = syn * 0.3 + ling * 0.5 + str * 0.2
Else
    simcom = syn * 0.4 + ling * 0.6
End If

End Function
```

1.2.4 Monge Elkan similarity metric

1.2.4.1 Melkan()

```
Function melkan(ByVal string0 As Range, ByVal string1 As Range, ByVal string2 As String, ByVal string3 As String) As Double
'declarations
Dim originals() As Variant
Dim successors() As Variant
Dim target As String
Dim successor As String
Dim tsuccessor As String
Dim counter As Double
Dim mkan As Double

'0 = range of query labels
'1 = range of query successors
'2 = target label
'3 = successor label

originals() = string0
successors() = string1
target = string2
tsuccessor = string3

counter = 1

'calculating Monge elkan similarity value by returning the highest match for each query label
For Each original In originals()

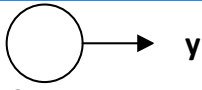
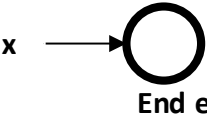

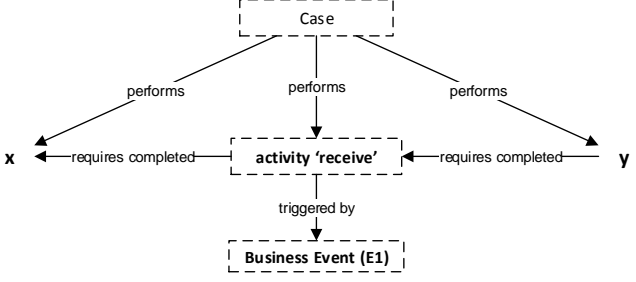
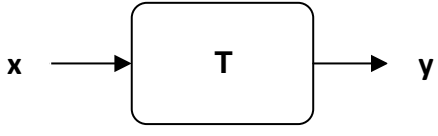
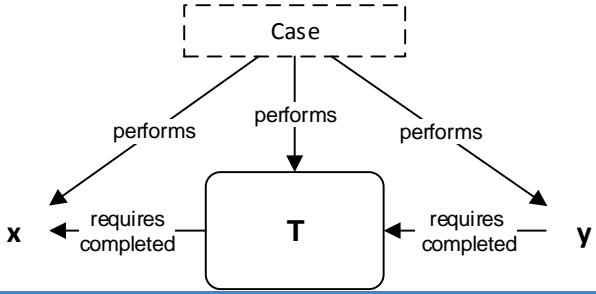

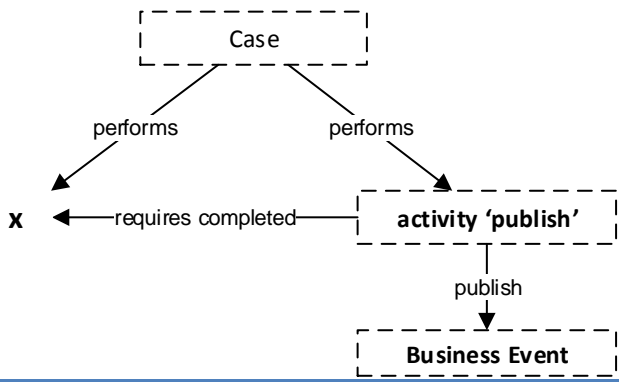
    successor = string1(counter, 1).Value
    mkan = Application.WorksheetFunction.Max(mkan, simcom(original, target, successor, tsuccessor))
    counter = counter + 1

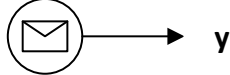
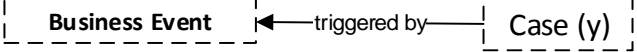
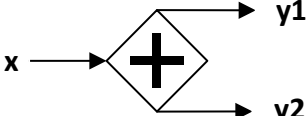
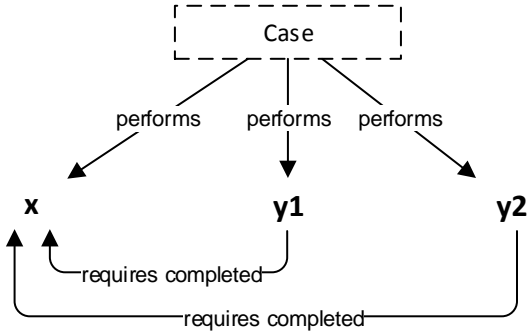
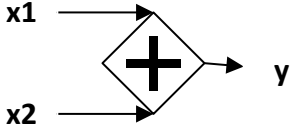
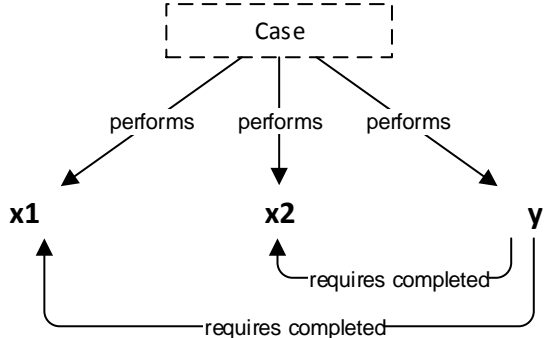
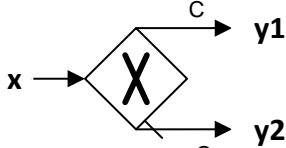
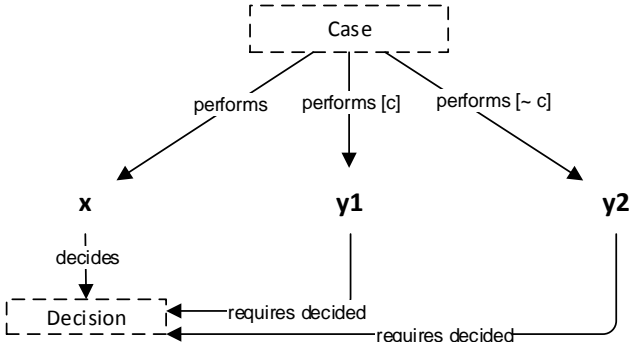
Next

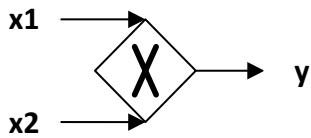
melkan = mkan
End Function
```

Appendix D: Well-formed transformation technique rules

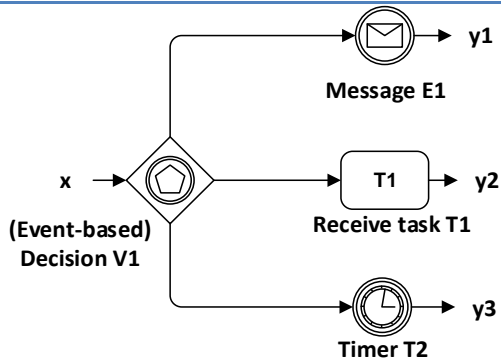
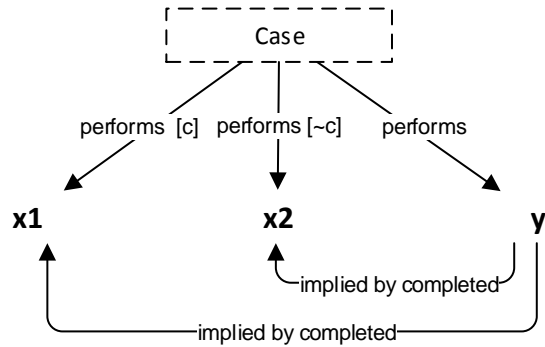
The table below contains all transformation rules for the well-formed transformation technique.

Source cluster	Target cluster
 <p>Start s</p>	Not applicable in DPMN. The starting point of a process in DPMN is not explicitly modeled.
 <p>End e</p>	Not applicable in DPMN. The end point of a process in DPMN is not explicitly modeled. Rather, the process is finished when all mandatory activities are completed.
 <p>Message E1</p>	
 <p>Task T</p>	
 <p>End message</p>	

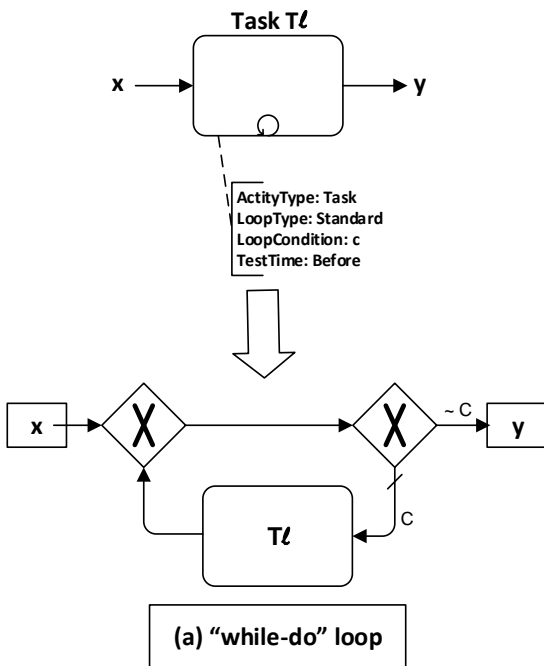
 <p>Start message</p>	
 <p>Fork F1</p>	
 <p>Join J1</p>	
 <p>(Data-based) Decision D1</p>	



Merge M1

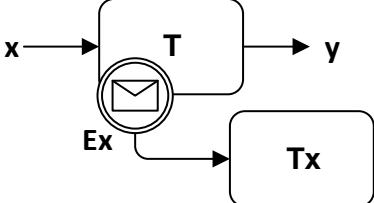
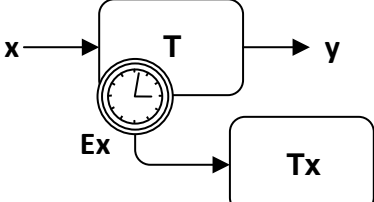
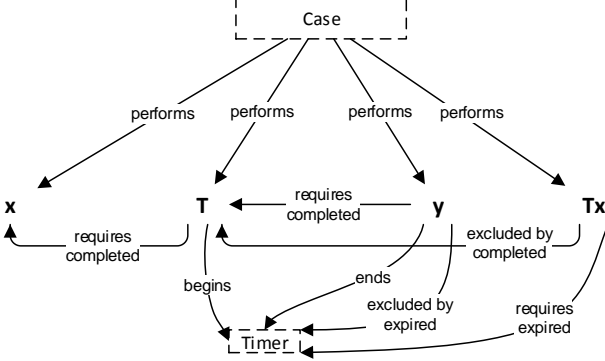
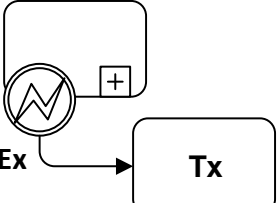
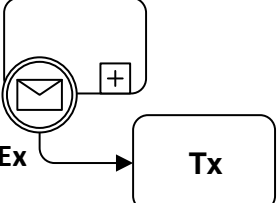
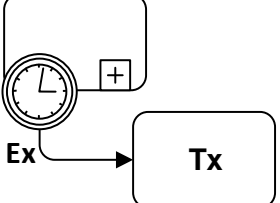


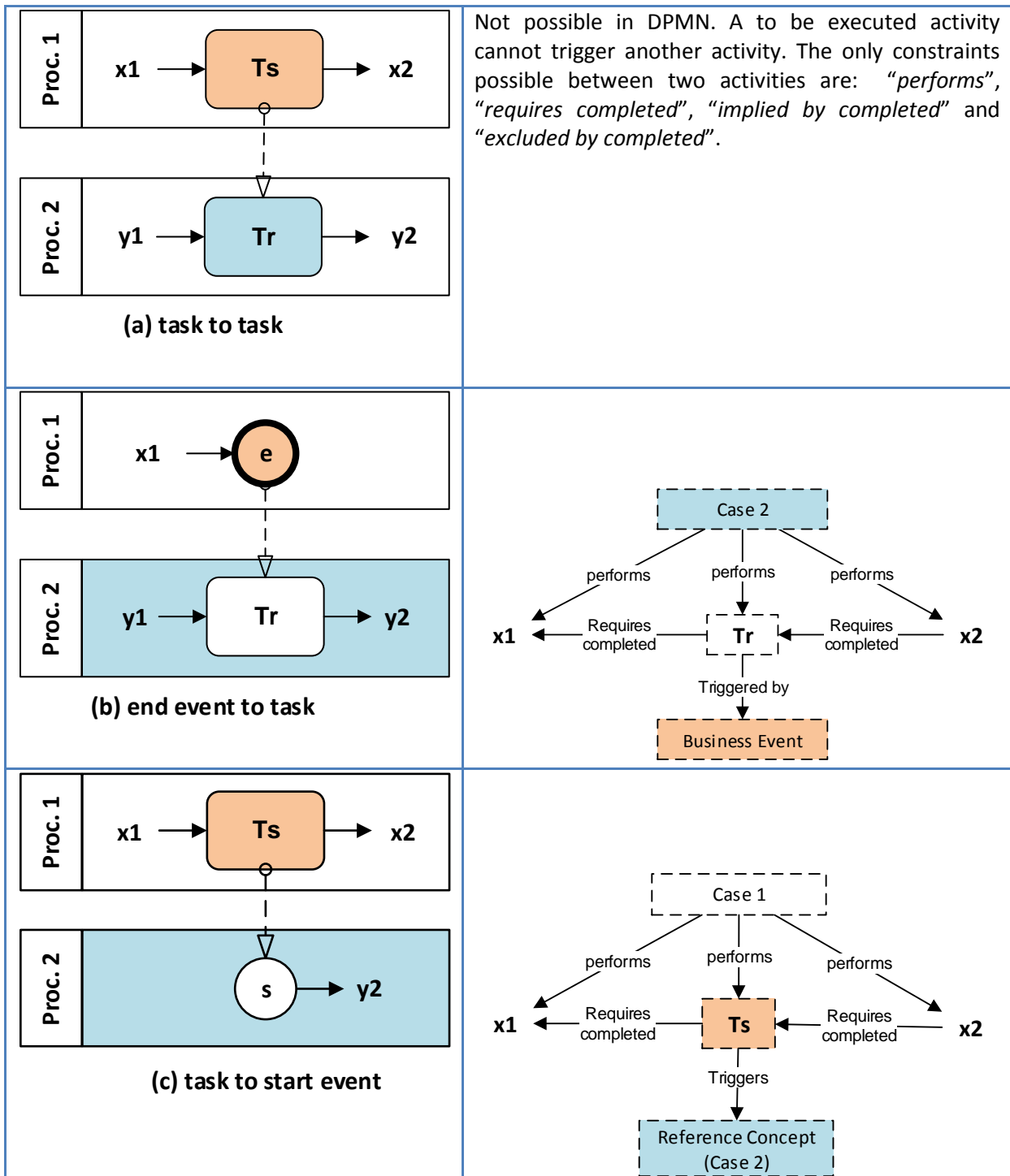
Not Possible in DPMN. An activity can be activated based on an event, but it is not possible to take a specific path based on an event that occurs. The paths have to be specified as either all mandatory or optional. In the first case, a process might wait for an event that never occurs. In the case of the latter, it is possible that no paths are executed at all.

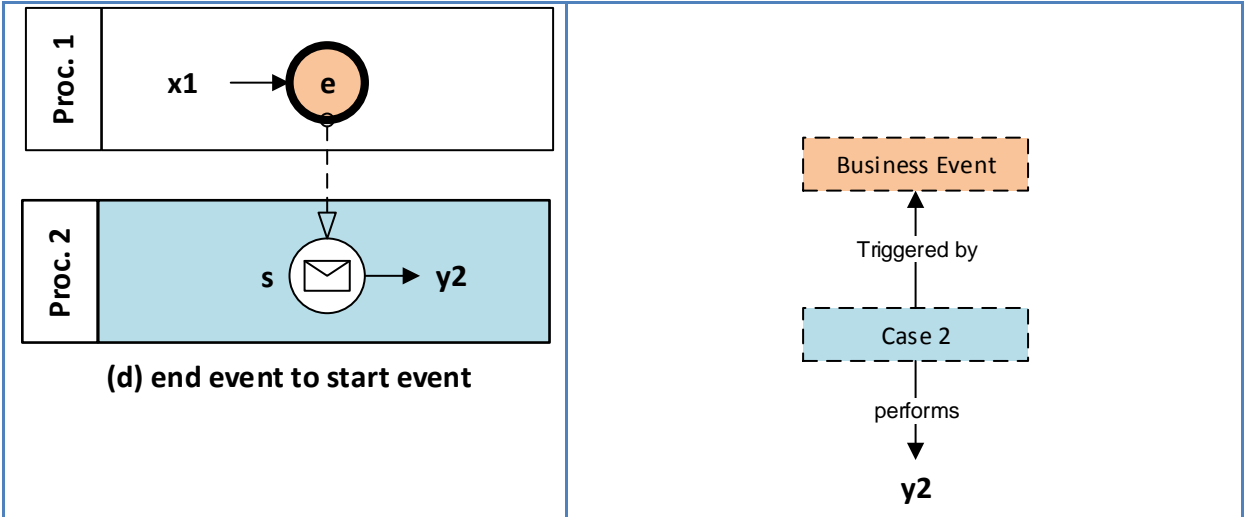


Not possible in DPMN. Activities cannot be performed multiple times in DPMN.

	<p>Not Possible in DPMN. Activities cannot be performed multiple times in DPMN.</p>
	<p>Not possible in DPMN. Activities cannot be performed multiple times in DPMN.</p>
	<p>Not possible in DPMN. A running task cannot be aborted by an error message.</p>

	<p>Not possible in DPMN. A running task cannot be aborted based on an incoming message.</p>
	
<p>Call subprocess P</p> 	<p>Not possible in DPMN. A sub-process in BPMN corresponds to a reference concept in DPMN. It is not possible to model outgoing relations from a reference concept, which makes an exception trigger from a reference concept impossible.</p>
<p>Call subprocess P</p> 	<p>Not possible in DPMN. A sub-process in BPMN corresponds to a reference concept in DPMN. It is not possible to model outgoing relations from a reference concept, which makes a message trigger from a reference concept impossible.</p>
<p>Call subprocess P</p> 	<p>Not possible in DPMN. A sub-process in BPMN corresponds to a reference concept in DPMN. It is not possible to model outgoing relations from a reference concept, which makes a timer trigger from a reference concept impossible.</p>





Appendix E: Graph productions

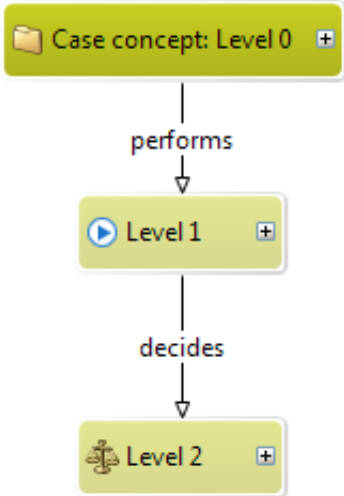
Production	Description
REFPROCESS	For each process model a case concept is created in the target DPMN model. The name of the process model corresponds to the name of the case concept.
REFPERFORMS	<p>When not explicitly stated otherwise, a concept is placed below the case concept. This corresponds to level 1 in Figure 24. The concept to be added is connected with a “<i>performs</i>”, “<i>creates</i>”, “<i>decides</i>”, “<i>triggers</i>” or “<i>classifies</i>” constraint, depending on the type of concept to be added. When a concept is placed as part of a preceding activity it is modeled on level 2, as depicted in Figure 24. The constraint can either be “<i>performs</i>”, “<i>creates</i>”, “<i>decides</i>”, “<i>triggers</i>” or “<i>classifies</i>”, depending on the type of concept to be added.</p> 
REFLABEL	Labels, which are annotations in the process model, correspond to note concepts in DPMN.
REFACTIVITY	An activity in BPMN corresponds to an activity in DPMN. In order to maintain the given order of activities. If an activity is preceded by another activity in the BPMN source model, a “ <i>requires completed</i> ” condition is placed between these activities in the DPMN model accordingly.
REFGATEWAY	This production is used transform gateway concepts. Because there are many different types of gateway concepts a decision tree has been created in order to determine how the gateway should be translated (Figure 25). The decision tree can only handle XOR and AND events, because no other gateway types were part of the sample set.

Figure 24: Hierarchy in the Declarative Process Modeling Notation

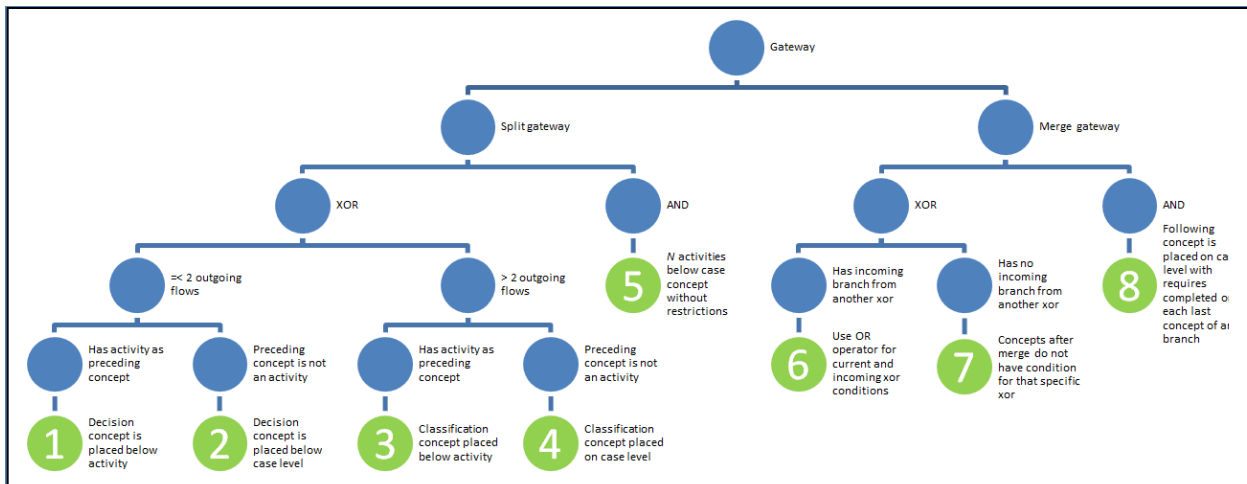


Figure 25: Decision tree for gateway concepts

1: A decision concept is created as part of the preceding activity (level 2). The label of the source gateway is used as a label for the decision concept. The concepts succeeding the gateway have conditions on their “*performs*” relation relating to the waypoint condition and are put on level 1.

2: A decision concept is created below the case concept. The concepts succeeding the gateway have conditions on their “*performs*” relation relating to the waypoint condition and are put on level 1.

3: A classification concept is created with classes conforming to the different outgoing waypoint conditions. The concept is placed as part of the preceding activity (level 2). The concepts succeeding the gateway have conditions on their “*performs*” relation relating to the waypoint condition and are put on level 1.

4: A classification concept is created with classes conforming to the different outgoing waypoint conditions. The concept is placed on level 1. The concepts succeeding the gateway have conditions on their “*performs*” relation relating to the waypoint condition and are put on level 1.

5: The concepts succeeding this gateway are put on level 1 without any conditions on their “*performs*” relation.

6: The merge gateway itself is not directly transformed into another concept. However, the concepts after the merge do not have a condition for that specific split anymore. The concept succeeding the merge has an “*implied by completed*” constraint on the last concept of each branch of the split. If the merge is accessed through an XOR from an earlier branch use the OR operator for specifying all conditions in the “*performs*” relation.

7: The merge gateway itself is not directly transformed into another concept. However, concepts after the merge do not have a condition for that specific split anymore. The concept succeeding the merge has an “*implied by completed*” on the last concept of each branch of the split.

8: The concept after an AND merge is placed on level 1. The concept has a “*requires completed*” constraint on the last concept of each AND branch.

REFEVENT

This production is used to transform concepts of the type event. A distinction is made between message events and non-message events. Furthermore, a distinction is made between intermediate, end and start events. The decision tree for event concepts can be seen in Figure 26.

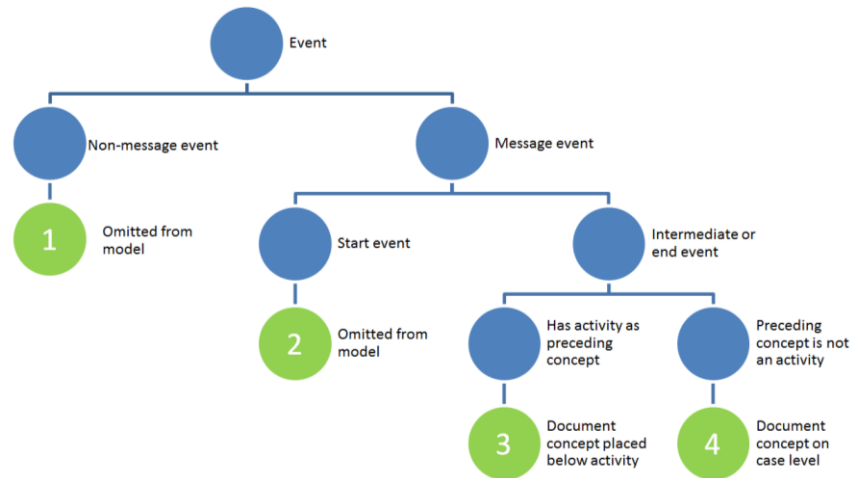


Figure 26: Decision tree for the event concept

1: If an event is not of the type “message” it is omitted from the target model.

2: If an event is not of the type “start” or “intermediate” it is omitted from the target model.

3: A document is created as part of the preceding activity (level 2).

4: A document is created and placed on level 1 and have a “requires completed” constraint on preceding concept.

Furthermore, an intermediate timer event cannot be modeled in DPMN. Therefore, if this concept is present in a source model the model is marked as not transformable.

REFSUBPROCESS

A sub-process in the source model is translated to a reference concept in the target model. When a sub-process is preceded by an activity, the reference concept is placed as part of the preceding activity with a “performs” relation (level 2). Otherwise, it is placed on level 1. Any succeeding concept is placed on level 1 with a “requires completed” constraint to the reference object.

REFPOOL

Pools are omitted from the target model.

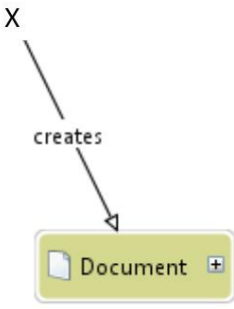
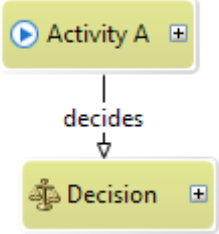
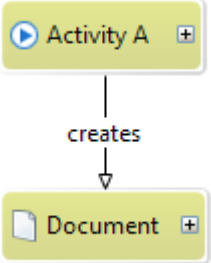
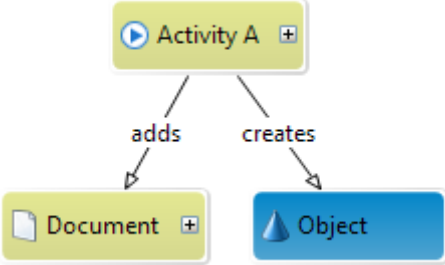
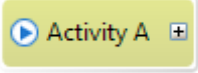
REFLANE

Lanes are omitted from the target model.

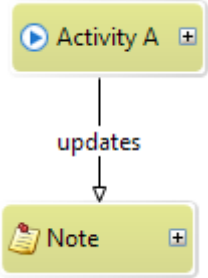
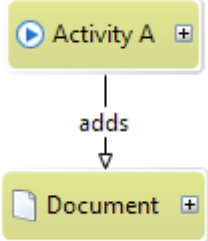
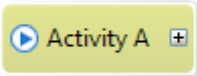
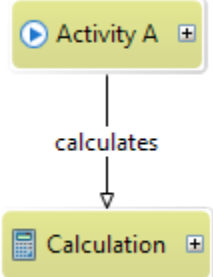
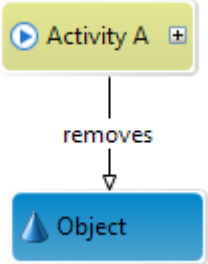
Appendix F: Verb classifications

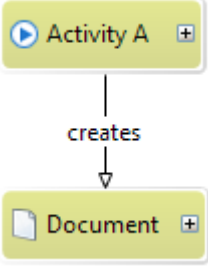
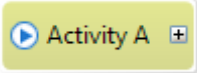
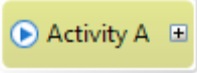
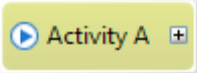
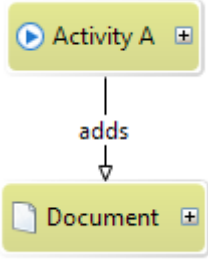
Main verb	Related verbs DPMN	Related verbs Case study data	Related verbs Mendling et al. (2011)
3. To determine	<ul style="list-style-type: none"> • Determine • Assess • Review • Verify • Validate • Decide • Check 	<ul style="list-style-type: none"> • Determine • Assess • Figure out • Verify • Analyze • Compare • Approve • Test 	<ul style="list-style-type: none"> • Assess • Decide
4. To create	<ul style="list-style-type: none"> • Create 	<ul style="list-style-type: none"> • Create • Make • Generate 	<ul style="list-style-type: none"> • Create
5. To complete	<ul style="list-style-type: none"> • Complete 	<ul style="list-style-type: none"> • Complement 	<ul style="list-style-type: none"> • Complete
6. To perform	<ul style="list-style-type: none"> • Perform • Process 	<ul style="list-style-type: none"> • Perform • Process • Run • Handle • Carry out 	<ul style="list-style-type: none"> • Process
7. To request	<ul style="list-style-type: none"> • Request 	<ul style="list-style-type: none"> • Request 	X
8. To respond	<ul style="list-style-type: none"> • Respond 	<ul style="list-style-type: none"> • Call • Inform 	<ul style="list-style-type: none"> • Communicate
9. To accept	<ul style="list-style-type: none"> • Accept 	<ul style="list-style-type: none"> • Receive • Back receive 	
10. To publish	<ul style="list-style-type: none"> • Publish • Submit • Offer 	<ul style="list-style-type: none"> • Send • Offer • Urge • Remind 	<ul style="list-style-type: none"> • Send
11. To report	<ul style="list-style-type: none"> • Report 	<ul style="list-style-type: none"> • Explain 	X
12. To update	X	<ul style="list-style-type: none"> • Update • Mutate • Correct • Convert 	<ul style="list-style-type: none"> • Modify • Transform
13. To archive	<ul style="list-style-type: none"> • Archive 	<ul style="list-style-type: none"> • Archive • Extend • Register • Record 	<ul style="list-style-type: none"> • Preserve
14. To collect	<ul style="list-style-type: none"> • Collect 	<ul style="list-style-type: none"> • Collect 	X
15. To calculate	X	<ul style="list-style-type: none"> • Calculate • Totalize 	X
16. To remove	X	X	<ul style="list-style-type: none"> • Remove • Destroy

Appendix G: Post-processing rules

Related cluster pair technique	
1.	Consecutive create and send activities. 
Semantic process similarity technique	
2.	Domain specific
Lexical analysis technique	
3.	To determine 
4.	To create 
5.	To complete 
6.	To perform 
7.	To request

		<pre> graph TD A[Activity A] -- creates --> B[Request Document] </pre>
8.	To respond	<pre> graph TD A[Activity A] </pre>
9.	To accept	<pre> graph TD A[Activity A] -- adds --> B[Document] </pre>
10.	To publish	<pre> graph TD A[Activity A] -- publishes --> B[Business Event] </pre>
11.	To report	<pre> graph TD A[Activity A] -- creates --> B[Note] </pre>

12	To update	 <pre> graph TD A[Activity A] -- updates --> B[Note] </pre>
13	To archive	 <pre> graph TD A[Activity A] -- adds --> B[Document] </pre>
14	To collect	 <pre> graph TD A[Activity A] </pre>
15	To calculate	 <pre> graph TD A[Activity A] -- calculates --> B[Calculation] </pre>
16	To remove	 <pre> graph TD A[Activity A] -- removes --> B[Object] </pre>
Machine learning technique		

17	Create document	 <pre> graph TD A[Activity A] -- creates --> D[Document] </pre>
18	Communicate activity	
19	Perform activity	
20	Update activity	
21	Register activity	 <pre> graph TD A[Activity A] -- adds --> D[Document] </pre>
22	Determine activity	