applied sciences

MDPI

Article

Transforming BPMN Processes to SBVR Process Rules with Deontic Modalities

Tomas Skersys [1,2,*], Paulius Danenas [1], Egle Mickeviciute [1] and Rimantas Butleris [1]

[1] Center of Information Systems Design Technologies, Kaunas University of Technology, K. Barsausko Str. 59, 51423 Kaunas, Lithuania
[2] Department of Information Systems, Kaunas University of Technology, Studentu Str. 50, 51368 Kaunas, Lithuania
* Correspondence: tomas.skersys@ktu.lt

Abstract: The Object Management Group (OMG) has put considerable effort into the standardization of various business modeling aspects within the context of model-driven systems development. Indeed, the Business Process Model and Notation (BPMN) is now arguably the most popular process modeling language. At the same time, the Semantics of Business Vocabulary and Business Rules (SBVR), which is a novel and formally sound standard for the specification of virtually any kind of knowledge using controlled natural language, is also gaining its grounds. Nonetheless, the integration between these two very much related standards remains weak. In this paper, we present one such integration effort, namely an approach for the extraction of SBVR process rules from BPMN processes. To accomplish this, we utilized model-to-model transformation technology, which is one of the core features of Model-Driven Architecture. At the core of the presented solution stands a set of model transformation rules and two algorithms specifying the formation of formally defined process rules from process models. Basic implementation aspects, together with the source code of the solution, are also presented in the paper. The experimental results acquired from the automatic model transformation have shown full compliance with the benchmark results and cover the entirety of the specified flow of work defined in the experimental process models. Following this, it is safe to conclude that the set of specified transformation rules and algorithms was sufficient for the given scope of the experiment, providing a solid background for the practical application and future developments of the solution.

Keywords: business process model; process rules; model transformation; SBVR; BPMN

check for updates

Citation: Skersys, T.; Danenas, P.; Mickeviciute, E.; Butleris, R. Transforming BPMN Processes to SBVR Process Rules with Deontic Modalities. *Appl. Sci.* **2022**, *12*, 8976. https://doi.org/10.3390/app12188976

Academic Editors: Malgorzata Pankowska and Emilio Insfran

Received: 23 July 2022
Accepted: 5 September 2022
Published: 7 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Despite all the efforts of OMG to make Business Process Model and Notation (BPMN) (OMG, 2013) a graphical language that could be read and understood equally by both IT and business people, the latter still lean toward the natural language specifications; this is especially true when it comes to documenting and validating business process models [1–4]. Better yet is to have both graphical and textual representations of business knowledge, because people might have different preferences towards process representation formats depending on the application purpose and the cognitive style of an individual himself [5,6]. This implies that graphical business process models should have unambiguous representation in natural language as well. In other words, there is an actual need for a solution that could support different kinds of visualizations (namely graphical and natural language-based) of business process models and beyond. Such a solution would be beneficial to all interested parties in any information systems development (ISD) effort, i.e., actual process participants who take part in real-world business activities, business analysts who model and analyze business processes, and system developers who develop computerized information systems supporting those business processes.

We put this business process model representation objective into a larger context, which is the development of solutions integrating various perspectives of business modeling and platform-independent modeling under the unified framework of OMG's Model-Driven Architecture (MDA) [7]. This is a long-term research initiative carried out by the researchers at Kaunas University of Technology (KTU) together with their respected partners in various research and development (R&D) projects, such as VeTIS ("Business Rule Solutions for Information Systems Development", 2007–2009), funded by the High Technology Development Program, Lithuanian State Science and Studies Foundation, and VEPSEM ("Integration of Business Processes and Business Rules on the Basis of Business Semantics", 2013–2015), funded by the European Social Fund (ESF). One of the main areas of our research in this domain is the integration of formally sound natural language-based business vocabularies and business rules with business processes. It is widely acknowledged that business processes and business rules are tightly related to each other. Yet, the actual integration of these two major aspects of business knowledge has not been explicitly defined by OMG in their respective standards, namely BPMN and Semantics of Business Vocabulary and Business Rules (SBVR) [8]. There is no direct referencing to SBVR, nor its concepts in any context in the latest version of BPMN. At the same time in SBVR, they mention business processes only once by saying that " ... a metamodel of a business process can import the SBVR XMI Metamodel package to relate processes to rules, or for modeling semantic formulations of rules that govern processes." It should be made clear that OMG does acknowledge the existence of direct or indirect relationships among business processes and other business modeling aspects, such as business rules models; however, these relationships are yet to be " ... defined more formally as BPMN and other specifications are advanced" [9]. In other words, the integration of OMG standards that define various aspects of business models, as well as information systems design, remains a prerogative of the international research community.

In this paper, we focus on one of the aspects of the integration of business processes with business vocabularies and business rules, specifically the transformation of BPMN processes to structured natural language-based process rules, which is a kind of behavioral business rules, expressed in SBVR. In that way, the natural language-based representation of business processes is well-aligned with the already existing business vocabularies and business rules of the same business domain. SBVR provides the means to properly structure and formalize business knowledge and to represent it in a natural language form, meaning the acquired SBVR process rules can be viewed and validated by people as well as computerized systems in the context of a single consolidated SBVR specification document. Such an augmented specification could then be used not only as a means to textually represent business processes but also for the requirements specifications, writing work instructions, generating software artifacts, improving the real-world business processes, and other purposes.

Further, the paper is organized as follows: an overview of the related work is presented in Section 2; Section 3 provides a specification of the transformation patterns, which is the core of our model transformation approach; basic implementation aspects of the approach are presented in Section 4, which is followed by an illustrative example of transforming BPMN process to a set of SBVR process rules presented in Section 5; Section 6 presents an overview and evaluation of the experiment results, and Section 7 explores threats to validity; finally, Section 8 concludes the presented research.

2. Related Work

2.1. State of the Art Research

Neither the integration of business processes with business rules nor the representation of business processes in natural language text (and vice versa) are novel ideas whatsoever. In model-driven ISD, the idea of integrating the two very much related aspects of business knowledge has been around for three decades, counting from the first notable publication

on this subject in 1991 [10]. Since then, numerous pieces of research have been conducted, providing a variety of integration approaches and applications [11–18].

At the same time, the representation of graphical business processes in natural language text has also gained significant attention. Most recently, Rosa et al., presented their approach for the user-assisted visual annotation of business process-related elements in textual documents [19]. The process model to text transformations (M2T) have gained ground in requirements engineering as well, e.g., Aysolmaz et al., generated well-structured system requirements documents from visual process models [20]. Meanwhile, others researched transformations in the opposite direction, that is, extracting process models from various textual documents [21–24]. In most of the overviewed related work, we observed one common issue with natural language texts, which is the inherent ambiguity and the lack of proper structuring of business knowledge expressed in natural language. In [25], the authors conclude that even the most advanced natural language processing (NLP) techniques still face many challenges when dealing with natural language texts in the domain of BPM; obviously, the same situation is inherent in the domain of ISD as well. In other words, without proper formalization, the representation of business knowledge in natural language presents a considerable challenge to the practical application of that business knowledge in BPM, ISD, and, presumably, other relevant domains.

We argue that OMG's Semantics of Business Vocabulary and Business Rules [8] provides all the required means to properly structure and formalize business knowledge while at the same time retaining the positive aspects provided by the natural language representation. In one of his interviews, Rob van Haarst, who is a renowned person in the business rules community, said that "SBVR is an extremely powerful method to formalize knowledge, forming a solid base for contracts, end-user documentation, work instructions, process design, and product development" [26]. Some go even further by stating that SBVR coupled with BPMN provides an optimal combination for expressing the semantics of business knowledge [27,28]. Moreover, SBVR specifications can be transformed into various conceptual modeling as well as software artifacts [29–32].

While there are many approaches researched on the integration of business processes and business rules, there is not much research on integrating specifically the concepts of BPMN and SBVR [33–35] and even less so on transforming BPMN processes to natural language-based SBVR business vocabularies and rules. Other than our own findings [36,37], the results of just a few other research pieces are to be found on the internet. In [36], we presented the SBVR and BPMN standards-based approach for the extraction of SBVR business vocabularies from BPMN process models; notably, only the business vocabulary part of the SBVR specification was considered in that research. In [37], certain founding principles for extracting SBVR process rules from the process models together with the experimentation results were presented; this research item lies in the fundament of the research work presented in this paper. Other than [37], only two other papers [38,39] published by the same authors presented some early research results describing the representation of BPMN processes in a form of SBVR process rules. However, these papers do not present a comprehensive transformation approach but rather the very basic principles and attempts to comprehend the problematics of such a transformation in general; a small set of basic illustrative model transformation rules were presented as well.

Several other research groups presented their approaches for transforming BPMN processes to business rules expressed in other than SBVR textual representation formats. The authors of [40] showed how compliance rules could be generated from the control flow of a BPMN process model. The acquired rules were aimed at the uniform representation of the activation of tasks using a specific human-readable language based on First-Order Logic (FOL). The presented approach provides informal complementation to a process model, whereas our approach deals with a more formal representation of business knowledge by the means of SBVR. In [41], an approach for the verification of BPMN processes against the rules defined as "IF ... THEN ... " expressions is presented. The approach uses the proposed Business Rules Language (BRL) to define structural patterns for the BPMN

process model, which can then be used to construct the BRL-compliant "IF ... THEN ... " verification rules for the specific process model. The paper lacks a better representation of the defined set of model patterns as well as an experimental investigation to prove the validity of the presented development. Compared with both [40,41], our model transformation approach handles a greater variety of relevant BPMN concepts and process model patterns, thus providing a more comprehensive representation of the process flow in a form of SBVR process rules.

Concerning our work, valuable results were presented by the researchers working with reverse transformations, that is, the automated development of BPMN process models based on the business knowledge stored in SBVR business vocabularies and business rules. The basis of all the reviewed approaches was a certain set of mapping rules for transforming SBVR business vocabularies and business rules into corresponding process model fragments composed of a single or a set of interrelated BPMN concepts forming certain structural process patterns [42–46]. When talking about process patterns, one can also consider the so-called recommender approaches for business process modeling [47]. Those recommendations may vary from suggesting presumably the most suitable label for the next activity being added to the sequence flow [48] up to recommending large process fragments [49] to the process model under development. Among other overviewed research items dealing with model transformations, the recommender approaches may also provide insights into the variety of possible structural business process patterns, which is a relevant subject to our approach.

2.2. Concepts' Definitions, Issues, and Pre-Conditions to Be Considered

Definition of a process rule. Since the first comprehensive research publications on the business rules subject in the last decade of the XX century [50–52], there has been a lot of arguing persisting about what business rules actually are or what they should be, etc. We will not go into the extensive discussion over this subject here, but rather introduce the concepts of a behavioral business rule and a process rule, the latter being one of the cornerstones of our model transformation approach presented in this paper.

In SBVR, a *behavioral business rule* indicates something business participants are either obliged to do (an obligation) or prohibited from doing (a prohibition), except where there is some explicit advice of permission given (a permission) [8]. We define a *process rule* (also known as a process-related rule) as a kind of behavior rule, which explicitly guides and constrains the flow of work itself (i.e., " ... the sequence and timing of activities" [53]) within a given business domain. In the context of BPMN and SBVR standards, process rules specify the flow of work presented in a BPMN process model in a well-structured set of natural language-based SBVR rules with deontic modalities inherent to the behavioral rules. Next to other kinds of business rules, the process rules have their place in the so-called process-rule continuum as well [13].

SBVR's capability to represent BPMN processes. When deciding on the matters of representing certain things, one must always make sure that the language selected for the task has sufficient expressive power to represent those things properly. In the case of SBVR representing concepts of BPMN, we concluded that SBVR should be extended to fully represent process rules and at the same time preserve the semantics of BPMN processes (especially if the reverse transformation is to be considered). Two extension options were considered. The first option was to extend the SBVR metamodel itself. Such an approach has already been applied by several other research groups to meet their specific requirements [33,42,54,55]. However, altering the metamodel of any widespread modeling standard comes at a cost, for example, tracking and applying changes on a metamodel level each time a new version of a standard is issued, accepting the high risk of closing-in in your own "sandbox" because the solution with the altered standard metamodel might not be accepted elsewhere, or limiting the range of tool support due to limited tool capabilities to support extensions on the metamodel level. The second option, which we consider an optimal solution for our approach, is using additional vocabulary describing BPMN

concepts. The peculiarity of SBVR is that SBVR itself is a business vocabulary that is meant to be imported into other vocabularies developed for any business domain. Importing other business vocabularies is a common practice in SBVR. Thus, the developed vocabulary for BPMN would be just another vocabulary to be imported for the process rules development; such an approach would not alter the SBVR metamodel, thus avoiding the "side-effects" of the previously described first option. However, it should be noted that the development of relevant SBVR business vocabularies for the specification of process rules is not within the scope of this paper, therefore this aspect will not be elaborated any further.

Linguistic Issues. Again, the linguistic aspect of business knowledge should be considered during the specification of SBVR business vocabularies, which is not within the scope of this paper, rather than during the phase of the specification of SBVR process rules. Nevertheless, a general awareness of the potential issues of this kind is a relevant topic in the context of this paper as well; thus, it will be briefly discussed in this subsection.

The generation of natural language texts from BPMN process models [1] and reverse transformation approaches [23] indicate that even advanced NLP techniques cannot guarantee the completeness and reliability of obtained results. Even though SBVR offers means to deal with synonyms, synonymous forms, homonyms, and some other peculiarities of a natural language, Ref. [56] showed that this is not sufficient even in the case of the English language, which has the largest NLP toolset available and is overall less complicated language compared with, for example, Lithuanian. SBVR's Structured English (SE) does not take into consideration declension, tenses, and plurality, which are the intrinsic features of natural languages. We argue that most of the aforementioned issues can be dealt with by introducing certain naming conventions to BPMN model elements and by following good modeling practices [36,37]. In that case, only a small set of simple NLP techniques is required to achieve satisfactory results.

3. Transformation of BPMN Processes to SBVR Process Rules

This section presents basic conceptual and engineering aspects of the approach for transforming BPMN processes to SBVR process rules with deontic modalities.

3.1. Transformation Scenario

The approach presented in this paper follows the so-called business rules "mantra" [57], which states that business rules are built on facts (or verb concepts, according to SBVR terminology), and facts are built on terms (noun concepts in SBVR terminology). Verb concepts and noun concepts form the basis of any business vocabulary; successively, one must have a business vocabulary to properly specify and manage business rules. This is a similar approach used in our other model transformation developments involving SBVR models [36,58].

The basic model transformation scenario (Figure 1) starts with the formation of the SBVR business vocabulary stage and then proceeds with the second stage, where SBVR business rules are being formed. Lastly, the overall SBVR model is validated by a business analyst or other interested party using the solution. This paper focuses on the second stage alone as the first stage was extensively presented in our previous papers mentioned above. Hence, we presume that a comprehensive SBVR business vocabulary holding both noun concepts and verb concepts (together with certain meta information underlying a BPMN process model) is already present and the formation of SBVR business rules may begin. Another reminder is that the transformations presented in this paper produce only a particular subset of SBVR business rules, namely process rules, leaving other kinds of business rules outside the scope of this paper.

The approach applies principles of a deterministic rule application strategy for model transformations according to the taxonomy presented in [59]. Following this taxonomy, our approach can be classified as a hybrid, bearing similarities from both declarative and imperative approaches, meaning some rules may require satisfying constraints for the presence of elements generated by other transformation rules. Indeed, the algorithm

presented in Figure 1 shows the hierarchical nature of such a transformation starting with "lower-level" concepts, such as noun concepts and verb concepts, then moving forward with the formation of business rules that are built upon those previously transformed concepts. Finally, following the model transformation taxonomy [60], we classify our transformation approach as an exogenous horizontal one-to-one model transformation.

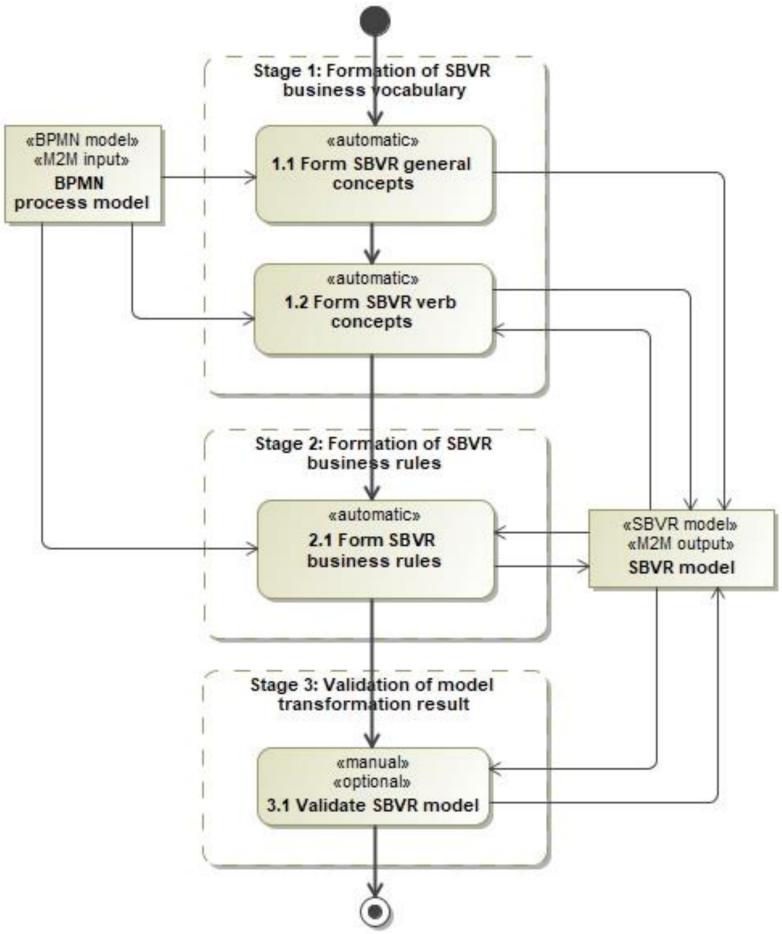


Figure 1. The overall architecture supporting the developed model transformation solution.

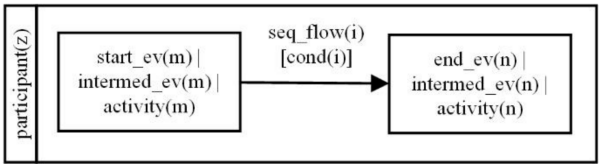
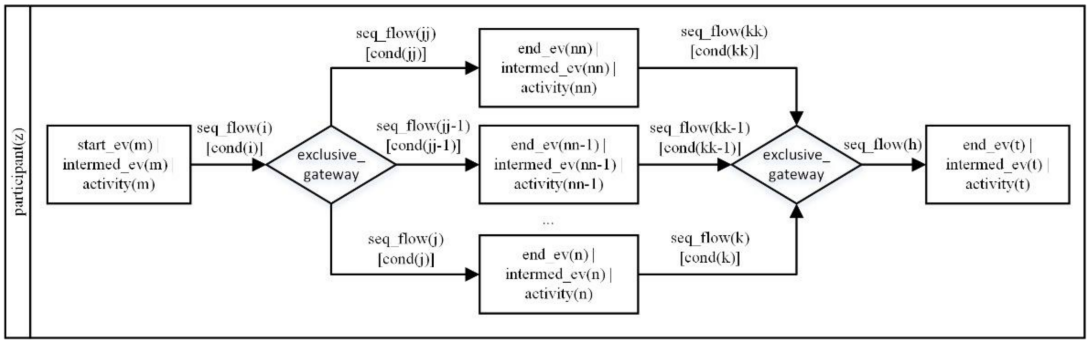
3.2. Transformation Rules

In general, a model transformation (T) represents a mapping between the input model and the output (target) models. For a specific model transformation rule (T_i) defined in our approach, the input is represented by a specific BPMN process model structure pattern (PP_i) and the output is represented by a specific SBVR process rule structure pattern (RP_i):

$$T_i: PP_i \rightarrow RP_i, \text{ where } i = 1, \ldots, 9$$

Further, in Table 1, we present model transformation rules for the automatic transformation of BPMN processes to SBVR process rules with deontic modalities. We state that the presented set of transformation rules is sufficient for the formation of a ruleset of SBVR process rules expressing the core flow concepts of a modeled BPMN process. Furthermore, experimenting with the acquired sets of process rules showed that those rulesets provide enough business knowledge to reverse engineer the flows of the initial processes; however, this is a subject for another research and will not be elaborated on further in this paper.

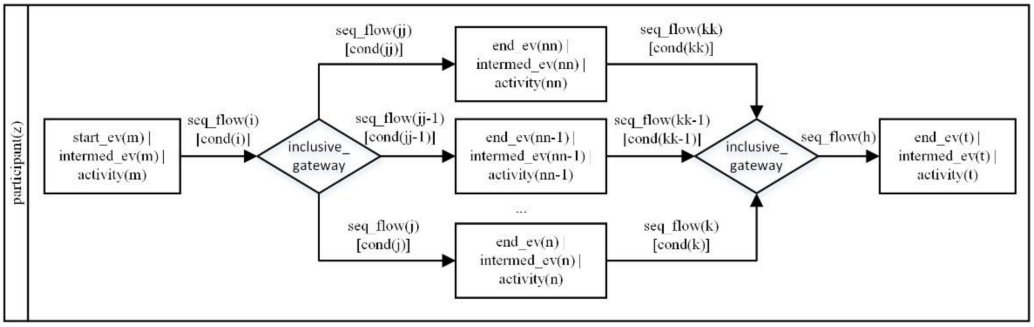
Table 1. Rules for the transformation of BPMN processes to SBVR process rules (T_i: $PP_i \rightarrow RP_i$).

T_i		Transformation Rule Definition
T_1	PP_1 [1]	Pattern with a sequence flow: 
	RP_1	(* RP1 *) "It is obligatory that ", (*vc(end_ev(n))* | *vc(intermed_ev(n))* | *vc(participant(z), activity(n))*), "after", (*vc(start_ev(m))* | *vc(intermed_ev(m))* | *vc(participant(z), activity(m))*), ["and if", *vc(cond(i))*], "."; (* /RP1 *)
	E.g.,	(* Next are examples of process rules acquired from the basic process pattern consisting of two sub-sequential flow objects*) It is obligatory that manager *registers* orderafter order *is received*. It is obligatory that order *is completed* after ordered production *is delivered*. It is obligatory that order *calculates* order estimate after manager *receives* registered order and if order information *is correct*.
T_2 [2]	PP_2	Pattern with diverging and | or converging exclusive gateway(s): 

	RP_2	(* RP2.div *)
		(* RULE PATTERN PART FOR A DIVERGING EXCLUSIVE GATEWAY: *)
		(* Repeat for every diverging sequence flow *seq_flow(j)* (where $j = j, \ldots, jj, n = n, \ldots, nn$) *)
		{
		(* If sequence flow *seq_flow(j)* contains condition *cond(j)* *)
		("It is obligatory that ", (*vc(end_ev(n))* | *vc(intermed_ev(n))* | *vc(participant(z), activity(n))*), "after",
		(*vc(start_ev(m))* | *vc(intermed_ev(m))* | *vc(participant(z), activity(m))*), " and if", *vc(cond(j))*, ["and", *vc(cond(i))*],
		".";) |
		(* If sequence flow *seq_flow(j)* does not contain condition *cond(j)* *)
		("It is permitted that", (*vc(end_ev(n))* | *vc(intermed_ev(n))* | *vc(participant(z), activity(n))*), "after",
		(*vc(start_ev(m))* | *vc(intermed_ev(m))* | *vc(participant(z), activity(m))*), ["and if", *vc(cond(i))*], ".";)
		}
		(* /RP2.div *)

		(* RP2.conv *) (* RULE PATTERN PART FOR A CONVERGING EXCLUSIVE GATEWAY: *) (* Next is an RP part for the first converging *seq_flow(k)* (where $k = k, n = n$) *) "It is obligatory that ", (*vc(end_ev(t))* | *vc(intermed_ev(t))* | *vc(participant(z), activity(t))*), "after", (*vc(intermed_ev(n))* | *vc(participant(z), activity(n))*), ["and if", *vc(cond(k))*], (* Next is an RP part for every next converging *seq_flow(k)* (where $k = k + 1, \ldots, kk, n = n + 1, \ldots, nn$) *) {"or", (*vc(intermed_ev(n))* | *vc(participant(z), activity(n))*, ["and if", *vc(cond(k))*]}, "."; (* /RP2.conv *)

Table 1. *Cont.*

T_i		Transformation Rule Definition
	E.g.,	(* Next is an example with two process rules acquired from two diverging branches of an exclusive gateway*) It is obligatory that manager *assigns* platinum order status after order *is registered*and if customer *is_a* platinum customer. It is obligatory that manager *assigns* normal order status after order *is registered*and if customer *is_a* first-comer customer. (* Next is an example with a converging exclusive gateway *) It is obligatory that order*is rejected* after manager *rejects* orderand if customer *is_a* first-comer customeror order confirmation *is past due*.
	PP_3	Pattern with diverging and | or converging inclusive gateway(s): 

T_3 [3]	RP_3	(* RP3.div *)
		(* RULE PATTERN PART FOR A DIVERGING INCLUSIVE GATEWAY: *)
		(* Repeat for every diverging sequence flow *seq_flow(j)* (where $j = j, \ldots, jj, n = n, \ldots, nn$) *)
		{
		"It is obligatory that ", (*vc(end_ev(n))* | *vc(intermed_ev(n))* | *vc(participant(z), activity(n))*), "after", (*vc(start_ev(m))* | *vc(intermed_ev(m))* | *vc(participant(z), activity(m))*), [" and if", *vc(cond(j))*], ["and", *vc(cond(i))*], ".";
		}
		(* /RP3.div *)

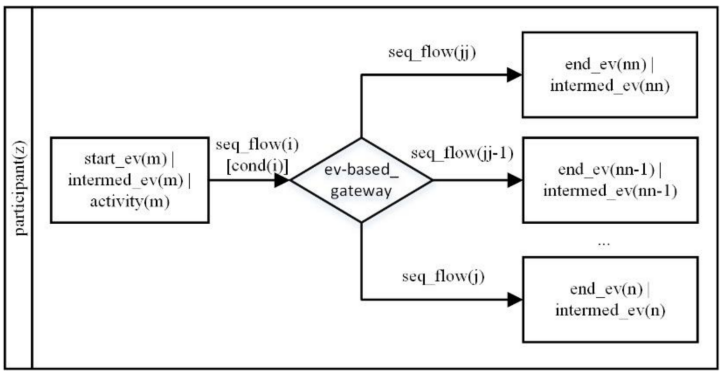
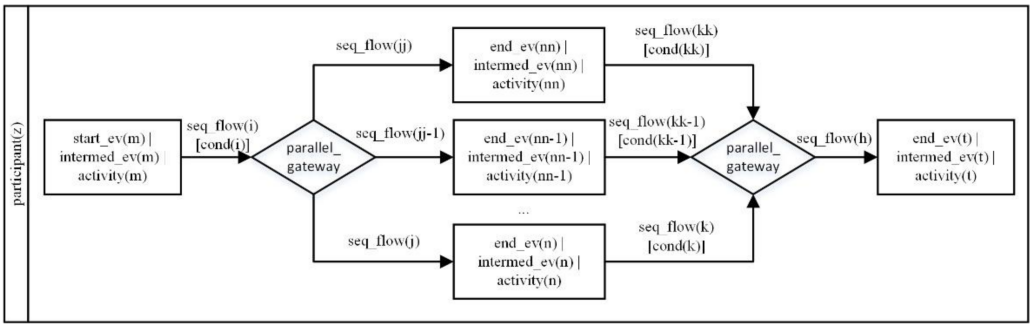
	RP_3	(* RP3.conv *) (* RULE PATTERN PART FOR A CONVERGING INCLUSIVE GATEWAY: *) "It is obligatory that", (*vc(end_ev(t))* | *vc(intermed_ev(t))* | *vc(participant(z), activity(t))*), " after", (* Next is an RP part for the first converging *seq_flow(k)* (where $j = j, k = k, n = n$) *) ["("],(*vc(vc(intermed_ev(n))* | *vc(participant(z), activity(n))*), ["if", *vc(cond(j))*], ["and", *vc(cond(k))*, ")"], (* Next is an RP part for every next converging *seq_flow(k)* (where $j = j + 1, k = k + 1, \ldots, kk, n = n + 1, \ldots, nn$) *) { "and", ["("], (*vc(intermed_ev(n))* | *vc(participant(z), activity(n))*), ["if", *vc(cond(j))*], ["and", *vc(cond(k))*, ")"] }, "."; (* /RP3.conv *)
	E.g.,	(* Next is an example with two process rules acquired from two diverging branches of an inclusive gateway*) It is obligatory that quality worker *attaches* quality certificate after ordered production*is packaged*. It is obligatory that expeditor*adds* extra packaging layer after ordered production*is packaged*and if ordered production *is a* fragile production. (* Next is an example with a converging inclusive gateway *) It is obligatory that expeditor*delivers* ordered productionafter quality worker *attaches* quality certificateand (expeditor*adds* extra packaging layer if ordered production *is a* fragile production).
T_4	PP_4	Pattern with diverging event-based gateway: 

Table 1. *Cont.*

T_i		Transformation Rule Definition

	RP_4	(* RP4 *)
		(* Repeat for every diverging sequence flow *seq_flow(j)* (where *j = j, . . . ,jj*) *)
		{
		"It is permitted that ", (*vc(end_ev(n))* | *vc(intermed_ev(n))*), "after", (*vc(start_ev(m))* | *vc(intermed_ev(m))* | *vc(participant(z), activity(m))*), [" and if", *vc(cond(i))*], ".";
		}
		(* /RP4 *)

	E.g.,	(* Next is an example with two process rules acquired from two diverging branches of an even-based gateway *) It is permitted that order confirmation *is received* after manager *sends* order confirmation request. It is permitted that order rejection *is received* after manager *sends* order confirmation request.
T_5	PP_5	Pattern with diverging and | or converging parallel gateway(s): 

	RP_5	(* RP5.div *)
		(* RULE PATTERN PART FOR A DIVERGING PARALLEL GATEWAY: *)
		(* Repeat for every diverging sequence flow *seq_flow(j)* (where *j = j, . . . ,jj, n = n, . . . ,nn*) *)
		{
		"It is obligatory that ", (*vc(intermed_ev(n))* | *vc(participant(z), activity(n))*), "after", (*vc(start_ev(m))* | *vc(intermed_ev(m))* | *vc(participant(z), activity(m))*), [" and if", *vc(cond(i))*], ".";
		}
		(* /RP5.div *)

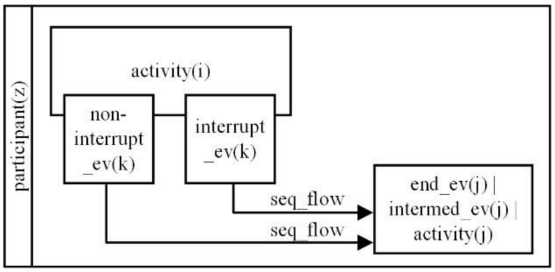
	RP_5	(* RP5.conv *) (* RULE PATTERN PART FOR A CONVERGING PARALLEL GATEWAY: *) "It is obligatory that ", (*vc(end_ev(t))* | *vc(intermed_ev(t))* | *vc(participant(z), activity(t))*), "after", (* Next is an RP part for the first converging *seq_flow(k)* (where *k = k, n = n*) *) ["("],(*vc(vc(intermed_ev(n))* | *vc(participant(z), activity(n))*), ["if", *vc(cond(k))*, ")"], (* Next is an RP part for every next converging *seq_flow(k)* (where *k = k + 1, . . . ,kk, n = n + 1, . . . ,nn*) *) { "and", ["("], (*vc(intermed_ev(n))* | *vc(participant(z), activity(n))*), ["if", *vc(cond(k))*,")"]], "."; (* /RP5.conv *)
	E.g.,	(* Next is an example with two process rules acquired from two diverging branches of a parallel gateway *) It is obligatory that quality worker *executes* quality checkafter ordered production*is produced*. It is obligatory that expeditor*schedules* order delivery dateafter ordered production*is produced*. (* Next is an example with a converging parallel gateway *) It is obligatory that order *is completed* after ordered production *is delivered*and order invoice *is fully covered*.
T_6	PP_6	Pattern with interrupting | non-interrupting boundary event: 
	RP_6	(* RP6 *) "It is permitted that ", (*vc(non-interrupt_ev(k))* | *vc(interrupt_ev(k))*), "when", *vc(participant(z), activity(i))*, "."; "It is obligatory that", (*vc(end_ev(j))* | *vc(intermed_ev(j))* | *vc(participant(z), activity(j))*), "after (", (*vc(non-interrupt_ev(k))* | *vc(interrupt_ev(k))*), "when", *vc(participant(z), activity(i))*,")."; (* ADDITIONAL RULE TO THE RULE PATTERN FOR AN INTERRUPTING BOUNDARY EVENT: *) ["It is prohibited that", *vc(participant(z), activity(i))*, "after", *vc(interrupt_ev(k))*, ".";] (* /RP6 *)

Table 1. *Cont.*

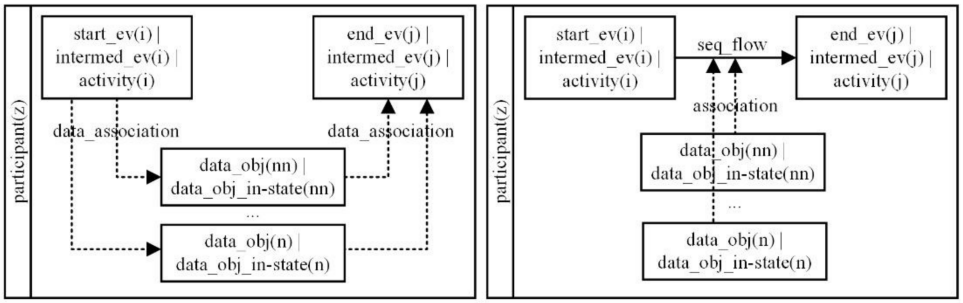
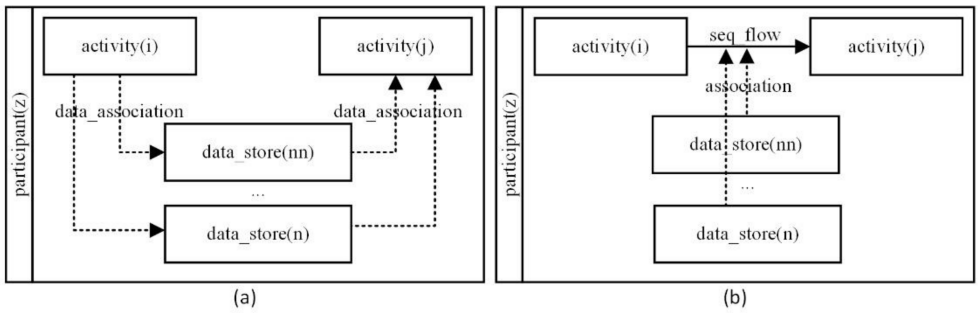
T_i		Transformation Rule Definition
	E.g.,	(* Next is an example with a set of process rules expressing behavior of an interrupting boundary event *) It is permitted that order cancellation*is received*when manager *registers* order. It is obligatory that manager *cancels* orderafter (order cancellation*is received*when manager *registers* order). It is prohibited that manager *registers* orderafter order cancellation*is received*.
T_7	PP_7	Pattern with two semantically equivalent variations of using data object with: (a) data association, and (b) association: 
	RP_7	(* RP7.out *) (* RULE PATTERN PART FOR AN OUTGOING DATA OBJECT (DATA OUTPUT): *) (* Next is an RP part for the first data object (data output) produced by an activity | event ($n = n$) *) "It is obligatory that ", (*gc(data_obj(n)) | gc(data_obj_in-state(n))*), "*is produced*", (* Next is an RP part for every next data object (data output) produced by an activity | event ($n = n + 1, \ldots, nn$) *) [{"and ", (*gc(data_obj(n)) | gc(data_obj_in-state(n))*), "*is produced*"}], "when", (*vc(start_ev(i)) | vc(intermed_ev(i)) | vc(participant(z), activity(i))*), "."; (* /RP7.out *)
		(* RP7.in *) (* RULE PATTERN PART FOR AN INCOMMING DATA OBJECT (DATA INPUT): *) (* Next is an RP part for the first data object (data input) required by an activity | event ($n = n$) *) "It is permitted that ", (*vc(end_ev(j)) | vc(intermed_ev(j)) | vc(participant(z), activity(j))*), " only if ", (*gc(data_obj(n)) | gc(data_obj_in-state(n))*), "*is provided to* ", *gc(participant(z))*, (* Next is an RP part forevery next data object (data input) required by an activity | event ($n = n + 1, \ldots, nn$) *) [{"and", (*gc(data_obj(n)) | gc(data_obj_in-state(n))*), "*is provided to*", *gc(participant(z))*}], "."; (* /RP7.in *)
	E.g.,	(* Next is an example with a produced (outgoing) data object *) It is obligatory that registered order *is produced* and production request *is produced*when manager *registers* order. (* Next is an example with a required (incoming) data object *) It is permitted that manager *calculates* order estimate only if registered order *is provided to* manager.
T_8	PP_8	Pattern with two semantically equivalent variations of using data stores with: (a) data association, and (b) association: 

Table 1. *Cont.*

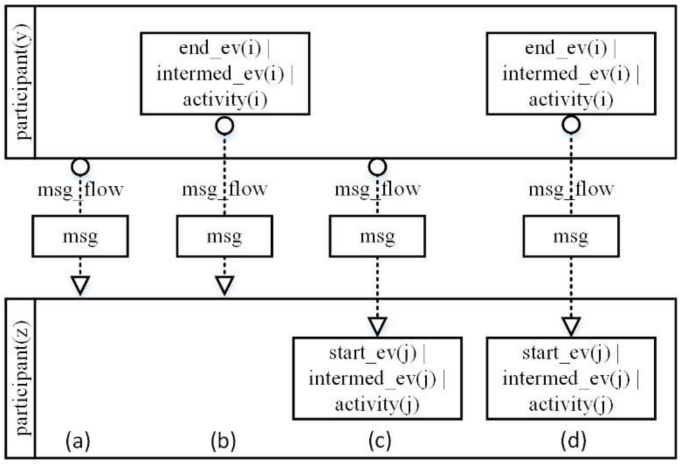
T_i		Transformation Rule Definition
	RP_8	(* RP8.out *) (* RULE PATTERN PART FOR PROVIDING DATA TO A DATA STORE: *) (* Next is an RP part for the first data store required by an activity ($n = n$) *) "It is obligatory that", *gc(data_store(n))*, "*is provided with data*", (* Next is an RP part for every next data store required by an activity ($n = n + 1, \ldots, nn$) *) [{"and", *gc(data_store(n))*, "*is provided with data*"}], "when", *vc(participant(z), activity(i))*, "."; (* /RP8.out *)
	RP_8	(* RP8.in *) (* RULE PATTERN PART FOR RECEIVING DATA FROM A DATA STORE: *) (*Next is an RP part for the first data store required by a particular activity ($n = n$) *) "It is permitted that", *vc(participant(z), activity(j))*, "only if", *gc(data_store(n))*, "*is available to*", *gc(participant(z))*, (* Next is an RP part for every next data store required by a particular activity ($n = n + 1, \ldots, nn$) *) [{"and", *gc(data_store(n))*, "*is available to*", *gc(participant(z))*}], "."; (* /RP8.in *)
	E.g.,	(* Next is an example with providing data to a data store *) It is obligatory that completed order reports *is provided with data* when manager *completes* order. (* Next is an example with a data store required to perform an activity *) It is permitted that manager *performs* quarterly sales assessment only if completed order reports *is available to* manager.
T_9	PP_9	Pattern comprised of the variations of "participant as a black-box" and "participant as a white-box" with incoming and outgoing message flows. Each process pattern *PP* is mapped to two rule patterns *RP* (one for a sending and one for a receiving end): 
	RP_9	(* RP9.Case(a)*) (* Next is the first pattern rule of Case (a) *) "It is permitted that", *gc(participant(y))*, "*sends*", ("*message*" | *gc(msg)*), "*to* ", *gc(participant(z))*, "."; (* The actual value of <something> in the expression "*sends <something> to*" depends on whether there is an actual message object attached to a message flow, or not; the same holds true for all other relevant pattern rules of *RP9* *) (* Next is the second pattern rule of Case (a) *) "It is permitted that", *gc(participant(z))*, "*receives* ", ("*message*" | *gc(msg)*), "*from* ", *gc(participant(y))*, ".";(* The actual value of <something> in the expression "*receives <something> from*" depends on whether there is an actual message object attached to a message flow, or not; the same holds true for all other relevant pattern rules of *RP9* *) (* /RP9.Case(a) *)
	RP_9	(* RP9.Case(b) *) (* Next is the first pattern rule of Case (b) *) ("It is obligatory that " | "It is permitted that "), *gc(participant(y))*, "*sends*", ("*message*" | *gc(msg)*), "*to*", *gc(participant(z))*, "when", (*vc(end_ev(i))* | *vc(intermed_ev(i))* | *vc(participant(y), activity(i))*), ".";(* The deontic formulation depends on the semantics of a specific source node the message flow is connected to (e.g., *Send Message Task* will imply obligation, while a *Manual Task* with an outgoing message flow attached to it – permission); the same holds true for the first pattern rule of the Case (d) of *RP9* *) (* Next is the second pattern rule of Case (b) *) "It is permitted that ", *gc(participant(z))*, "*receives*", ("*message*" | *gc(msg)*), "*from*", *gc(participant(y))*, "."; (* /RP9.Case(b) *)

Table 1. *Cont.*

T_i	Transformation Rule Definition
	(* RP9.Case(c) *) (* Next is the first pattern rule of Case (c) *) "It is permitted that ", *gc(participant(y))*, "*sends* ", ("*message*" | *gc(msg)*), "*to* ", *gc(participant(z))*, "."; (* Next is the second pattern rule of Case (c) *) (* Next is a rule pattern variation when a receiving node is an event *) ("It is obligatory that ", (*vc(start_ev(j))* | *vc(intermed_ev(j))*), "when", *gc(participant(z))*, "*receives*", ("*message*" | *gc(msg)*), "*from*", *gc(participant(y))*) | (* Next is a rule pattern variation when a receiving node is an activity *) ("It is obligatory that" | "It is permitted that", *gc(participant(z))*, "*receives*", ("*message*" | *gc(msg)*), "*from*", *gc(participant(y))*, "when", *vc(participant(z), activity(j))*), "."; (* The deontic formulation for both rule variations depends on the semantics of a specific target node the message flow is connected to (e.g., *Receive Message Task* will always imply obligation, while a *Manual Task* with an incoming message flow attached to it – permission); the same holds true for the first pattern rule of the Case (d) of *RP9* *) (* /RP9.Case(c) *)
	(* RP9.Case(d) *) (* Next is the first pattern rule of Case (d) *) ("It is obligatory that " | "It is permitted that"), *gc(participant(y))*, "*sends* ", ("*message*" | *gc(msg)*), "*to*", *gc(participant(z))*, "when", (*vc(end_ev(i))* | *vc(intermed_ev(i))* | *vc(participant(y), activity(i))*), "."; (* Next is the second pattern rule of Case (d) *) (* Next is a rule pattern variation when a receiving node is an event *) (" It is obligatory that", *vc(start_ev(j))* | *vc(intermed_ev(j))*, "when", (*gc(participant(z))*, "*receives*", ("*message*" | *gc(msg)*), "*from* ", *gc(participant(y))*) | (* Next is a rule pattern when a receiving node is an activity*) ("It is obligatory that " | "It is permitted that", (*gc(participant(z))*, "*receives*", ("*message*" | *gc(msg)*), "*from*", *gc(participant(y))*, "when", *vc(participant(z), activity(j))*), "."; (* /RP9.Case(d) *)
E.g.,	(* Next is an example of the first rule of Case (a) *) It is permitted that customer *sends* order *to* manager. (* Next is an example of the second rule of Case (a) *) It is permitted that manager *receives* order *from* customer. (* Next is an example of the first rule of Case (b). The sending node is a send message event, which implies an obligation to act *) It is obligatory that customer *sends* order *to* managerwhen order request *is sent*. (* Next is an example of the second rule of Case (b) *) It is permitted that manager *receives* order *from*customer. (* Next is an example of the first rule of Case (c) *) It is permitted that customer *sends* order estimate confirmation *to* manager. (* Next is an example of the second rule of Case (c). The receiving node is a receive message event, which implies an obligation for an even to happen when a message is received *) It is obligatory that order estimate *is confirmed* when manager *receives* order estimate confirmation *from* customer. (* Next is an example of the second rule of Case (c). The receiving node is a manual task, which implies a permission to act *) It is permitted that manager *receives* order estimate confirmation *from* customerwhen manager *schedules* order delivery date. (* Next is an example of the first rule of Case (d) *) It is obligatory that customer *sends message to* manager when order estimate *is confirmed*. (* Next is an example of the second rule of Case (d). Note that the information about the message being received is stated implicitly, which naturally brings a certain level of ambiguity to the specification, especially when there is a number of messages being sent within a modeled process; therefore, it is advised to avoid implicit messages whenever possible, and this is one of the good practices for modeling processes *) It is obligatory that confirmed order estimate *is received* when manager *receives message from* customer.

[1] Note that in patterns PP_i, most of the listed BPMN concepts are supertypes, meaning that those elements generalize sets of subtypes. For example, in PP_1, the identified *activity* represents a BPMN concept *Activity*, which in BPMN is defined as a *Process* step, e.g., *Task*, *Sub-Process*; or the *intermediate_event* representing a BPMN *Intermediate Event*, which in BPMN has an extensive set of subtypes, e.g., *Message Intermediate Event*, *Timer Intermediate Event*. Such pattern modeling decision was made to reduce the representational complexity of PP_i, thus avoiding cluttering the visual representation with the excessive level of detail. [2] The pattern T_2 also holds several additional variations to handle specific cases of an exclusive gateway, which are not presented in this paper. The specific cases include handling of *default* sequence flows that hold no condition expressions (those are handled through formulating joins of negations of the conditions of all the other sequence flows diverging from that exclusive gateway) or exclusive gateways with no condition expressions on any of the diverging sequence flows (which is considered a bad modeling practice but still used quite often). [3] The pattern T_3 holds two process patterns PP_i resulting in the same rule pattern RP_3: (1) the one presented in the paper with a set of *sequence flows* diverging from an *inclusive gateway*; (2) the second one with a set of two or more *conditioned sequence flows* diverging from *activity(i)*, which is an alternative to a diverging inclusive gateway. These two cases were merged into one pattern for simplicity reasons.

For conceptual clarity, we specified all transformation rules using a unified template, the main principles of which were taken from [58]. In each transformation rule, the PP_i is presented as a fragment of a conceptualized BPMN process diagram, and the RP_i as a formalized textual structure expressed in Extended Backus–Naur form (EBNF). In PP_i, some element abbreviations are used; however, with basic knowledge of BPMN, those abbreviated elements are easily identified intuitively (e.g., *start_ev* stands for *Start Event* or *seq_flow* for *Sequence Flow*). In the formalization of RP_i, the *vc(a)* refers to a particular SBVR noun concept representing a particular BPMN concept; in its turn, the *vc(a, b)* refers to a particular SBVR verb concept constructed upon a pair of noun concepts representing particular concepts of the BPMN process model.

Again, as was mentioned before, both noun concepts and verb concepts are formed during the execution of Stage 1 of the model transformation process (Figure 1), which is outside the scope of this paper. In addition, it is important to note that the presented set of transformation rules covers a subset of concepts of the BPMN process model relevant to the extraction of the *process* rules, meaning that the BPMN concepts that are unrelated to the extraction of the process rules or are uninterpretable by the automated interpreters (e.g., *Complex Gateway*), are left out of the scope of the current version of the approach as well. The following subset of BPMN concepts was considered in this transformation approach: pool/lane, start/intermediate/end events, interrupting/non-interrupting boundary events, activity, exclusive/inclusive/event-based/parallel gateways, data object, data store, message, message flow, sequence flow, and sequence flow with a condition.

In Table 1, each formalized rule T_i is followed by examples of SBVR process rules as the execution result of that specific transformation rule. For more examples of the actual instantiation of the specified transformation rules refer to Section 5.

Full semantic expressiveness of the acquired process rules is reached when considering the underlying business vocabulary comprising noun concepts and verb concepts because those concepts hold all the relevant semantic information inherited from the BPMN process model from which they were acquired (e.g., SBVR general concepts being enriched with descriptions and meta-information about the types of BPMN concepts that those general concepts represent). Furthermore, this aspect becomes critical when it comes to reverse engineering BPMN process models from SBVR specifications if there would be such a need.

3.3. Extracting Process Rules from Multiple Cascading Gateways

Other than the defined process patterns, our approach also supports the so-called cascading gateways, that is, when a sequence flow leaves one diverging/converging gateway and leads directly to another diverging/converging gateway forming a cascade of two sub-sequentially diverging gateways. The processing of cascading gateways follows the so-called backward-chaining principle analyzing the required number of flow objects retrospectively and combining multiple conditions of the diverging and converging sequence flows into one process rule. The processing of such chains can be challenging at both the conceptual and engineering levels.

To present the developed algorithm, let us first introduce the notions of *activity node neighborhood* (Definition 1).

Definition 1. *Activity node neighborhood is defined as a tuple*

$$A_N = \left\langle act,\ S_{act},\ SF_{inc},\ SF_{out},\ SF^D_{inc}, SF^D_{out} \right\rangle$$

where:

- *activity is an activity node: (type(activity)∈ {Task, Event, SubProcess};*
- S_{act} *is a set of subjects performing the defined activity;*
- SF_{inc} *is a set of sequence flows incoming to the activity act, with or without conditions;*
- SF_{out} *is a set of sequence flows outgoing from the activity, with or without conditions;*
- SF^D_{inc} *is a set of default sequence flows incoming to the activity;*

- SF^{D}_{out} *is a set of default sequence flows outgoing from the activity.*

Given such a formal structure, one can apply it to process the defined rules T_2, T_3, T_4, and T_5. To process a chain of gateways, one must also consider advanced structures supporting recursive gateway processing. Therefore, a *gateway neighborhood* definition must be introduced (Definition 2).

Definition 2. *Gateway neighborhood is defined as a tuple*

$$G_N = \left\langle gate,\ type,\ S_{activity},\ \mathcal{A}_{Ninc}\ ,\ \mathcal{A}_{Nout},\ ,\ \mathcal{G}_{Ninc},\ ,\ \mathcal{G}_{Nout}\ ,\ SF_{inc},\ SF_{out},\ SF^{D}_{inc},\ SF^{D}_{out},\ R_{part} \right\rangle$$

where:

- *gate is a BPMN gateway node;*
- *type is a type of the defined gateway gate (type∈ {inclusive, exclusive, event-based, parallel});*
- $\mathcal{A}_{Ninc}$ *is a set of activities incoming to the gateway gate;*
- $\mathcal{A}_{Nout}$ *is a set of activities outgoing from the gateway gate;*
- $\mathcal{G}_{Ninc}$ *is a set of other gateways incoming to the gateway gate;*
- $\mathcal{G}_{Nout}$ *is a set of other gateways outgoing from the gateway gate;*
- SF_{inc} *is a set of incoming to the gateway gate, with or without conditions;*
- SF_{out} *is a set of outgoing from the gateway gate, with or without conditions;*
- SF^{D}_{inc} *is a set of default sequence flows incoming to the gateway gate;*
- SF^{D}_{out} *is a set of default sequence flows outgoing from the gateway gate;*
- R_{part} *is a partial rule, recursively extracted at the gateway gate.*

Gateway neighborhoods are constructed recursively and exclusively from incoming or outgoing sequence flows, which have gateways at their ends. For our backward-chaining algorithm, we will use only $\mathcal{G}_{Ninc}$; then, gateway element *el* is extracted as follows:

$$\mathcal{G}^{el}_{N_{inc}} = \mathcal{G}^{el}_{N_{inc}} \cup G_N(g),\ \forall g \in \{source(SF_{inc}) |\ source(SF_{inc})\ is\ gateway\ and\ target(SF_{inc} = el\}$$

where SF_{inc} is a set of BPMN sequence flows incoming to *el*.

Further, we present an algorithm (Algorithm 1), which is used to extract process rules from tuples of sub-sequentially interconnected gateways.

Algorithm 1: extractComplexRules

Input: BPMN element *el*: *type(el) ∈ {Task, Event, SubProcess}*
$A^{el}_N \leftarrow$ *activity node neighbourhood for el*
$SF^{g}_{inc} \leftarrow \{source(SF_{inc}) | source(SF_{inc})\ is\ gateway\ and\ target(SF_{inc}) = el\}$
If $SF^{g}_{inc} = \varnothing$:
    Run extract T_1 *rule;* ***Return*** $\varnothing$
$FinalRules \leftarrow \varnothing$
For $SF \in SF^{g}_{inc}$:
    $G^{g}_{N} \leftarrow G_N\ for\ gateway\ in\ source(SF)$
    $createPartialRule(G^{g}_{N})$
        $FinalRules \leftarrow FinalRules \cup concat\left(extractRule\left(A^{el}_N\right),\ G^{g}_{N}.R_{part}\right)$
End for
Output: a set of extracted rules *FinalRules*.

The procedure *createPartialRule* is based on the detection of the so-called *boundary gateway.* We define a gateway as a *boundary* if it does not contain any incoming sequence flows with other gateways as their sources ($\mathcal{G}_{Ninc}$ is empty). For boundary gateways, rules T_2, T_3, T_4, and T_5 can be directly applied to extract partial rules that are further combined recursively. The pseudocode for this procedure is provided in Algorithm 2.

Algorithm 2: createPartialRule

Input: *gateway neighborhood* G_N.
If $G_N.\mathcal{G}_{Ninc} \leftarrow \varnothing$:
  $G_N.R_{part} \leftarrow extractSimpleRule(G_N)$
  Return G_N
$G_N.R_{part} \leftarrow \varnothing$
$defaultCond \leftarrow \varnothing$
 # First process gateways
For $G_N^i \in G_N.\ \mathcal{G}_{Ninc}$
  $createPartialRule(G_N^i)$
    $cond_{partial} \leftarrow$ *create partial antecedent from conditions in* $G_N^i.SF_{inc}$
    if $G_N^i.gate \in \{el | el = source(SF),\ \forall SF \in SF_{inc}^D\}$
      $defaultCond \leftarrow defaultCond \cup cond_{partial}$
    else
      $G_N.R_{part} \leftarrow concat(G_N^i.R_{part}^i,\ 'IF', cond_{partial})$
End for
 # Then process activities
For $A_N^i \in G_N.\ \mathcal{A}_{Ninc}$
    $cond_{partial} \leftarrow$ *create partial antecedent from conditions in* $A_N^i.SF_{inc}$
    if $A_N^i.act \in \{el | el = source(SF),\ \forall SF \in SF_{inc}^D\}$
      $defaultCond \leftarrow defaultCond \cup cond_{partial}$
  else
      $G_N.R_{part} \leftarrow concat(G_N.R_{part},\ 'IF',\ cond_{partial})$
End for
 # Finally, add default conditions
If $|defaultCond| \neq \varnothing$:
  $G_N.R_{part} \leftarrow concat(G_N.R_{part},\ 'OTHERWISE')$
  For $cond \in defaultCond$
    $G_N.R_{part} \leftarrow concat(G_N.R_{part},\ cond)$
  End for
Output: updated gateway neighborhood G_N.

The developed approach is capable of processing chains composed of a virtually unlimited number of sub-sequential gateways. However, one should consider good modeling practices stating that modeling decisions via multiple cascading gateways should be avoided as this results in over-complex process rules, which in turn may result in poor readability and understandability of the presented business process logic by humans.

To illustrate the presented algorithms, we will use the example from Section 5. Let us consider the following process rule:

> It is obligatory that order request *is rejected*after provider *analyzes* orderand (provider *evaluates* extra conditionsif order *has* extra conditions) and if order *cannot be fulfilled*.

The formation of this rule starts from the event *order request is rejected*. This element has an incoming sequence flow and a gateway element at the start of the chain under consideration (i.e., the source element), therefore we can identify the instance gateway neighborhood. Yet, this is not a boundary gateway as it has a sequence flow incoming from another preceding gateway. In turn, that preceding gateway is connected with the tasks *analyze order* and *evaluate extra conditions* acting as source elements for the connecting sequence flows. Hence, we move to this gateway as it has no directly connecting preceding gateways and mark it as a boundary gateway. As mentioned, this gateway has two incoming task elements, namely, *analyze order* and *evaluate extra conditions*. These tasks are used to create partial antecedent conditions provider *analyzes* order and (provider *evaluates* extra conditionsif order *has* extra conditions) with the latter "if ... " formulation taken from the guard condition of the sequence flow incoming to the task *evaluate extra conditions*. Finally, we append the guard condition *order cannot be fulfilled* from the sequence flow connecting to the event *order request is rejected* with the gateway element that represents the condition required to trigger this event. Internally, these elements are assigned to the gateway neighborhood structures, which are then used to form our process rule.

4. Implementation

This section describes the prototype for transforming BPMN processes to SBVR process rules. The latest version of the plugin prototype and other useful information is available at https://bitbucket.org/pauliusdan/sbvr-extraction (accessed on 29 August 2022).

The implementation architecture (Figure 2) comprises a CASE system and four other systems entirely or partially developed by our team during various R&D projects:

- MagicDraw is a CASE system supporting the newest official versions of UML, BPMN, and some other OMG modeling standards. However, it does not natively support any version of SBVR, which is extensively used in our approach. Yet, MagicDraw has Open Java API, enabling the development of custom plug-ins. It also has other advanced capabilities for UML profiling, analysis, and reporting, as well as domain-specific language (DSL) development.
- SBVR modeler plug-in for MagicDraw, which uses a UML profile for SBVR. Both the plug-in and the profile were developed during the VEPSEM project, in which the authors of this paper also took part. In the CASE system, SBVR business vocabulary is graphically represented as a set of so-called fact diagrams. The plug-in also supports the specification of SBVR business rules as formal text-based expressions.
- External SBVR editor VeTIS is a tool developed during another R&D project, VeTIS. The editor supports the development and syntactic validation of SBVR business vocabularies and business rules expressed as well-structured natural language-based textual specifications. It supports various additional properties of SBVR concepts defined in the SBVR standard and is also capable of augmenting business vocabulary entries with complementary information, such as synonyms and definitions for general concepts from WordNet.
- Tool for BPMN process transformation to SBVR process rules is also implemented as a plug-in for the MagicDraw and is entirely developed by the authors of this paper. The tool implements both automatic model-to-text (to be used in the VeTIS editor) and model-to-model transformations, which may be called out selectively from the context menu of the CASE system's modeling environment. The specified model transformation rules were implemented using the QVT model transformation language [61]. We consider the model transformation component to be of technology readiness level 4 (TRL 4) (the referred technology readiness levels are defined in the HORIZON 2020 documentation and can be accessed at: https://ec.europa.eu/research/participants/data/ref/h2020/wp/2014_2015/annexes/h2020-wp1415-annex-g-trl_en.pdf (accessed on 29 August 2022)), meaning this development was tested in the laboratory environment.
- Model integration plug-in provides model integration capability and QVT transformation engine basis. This plug-in was developed during the VEPSEM project.

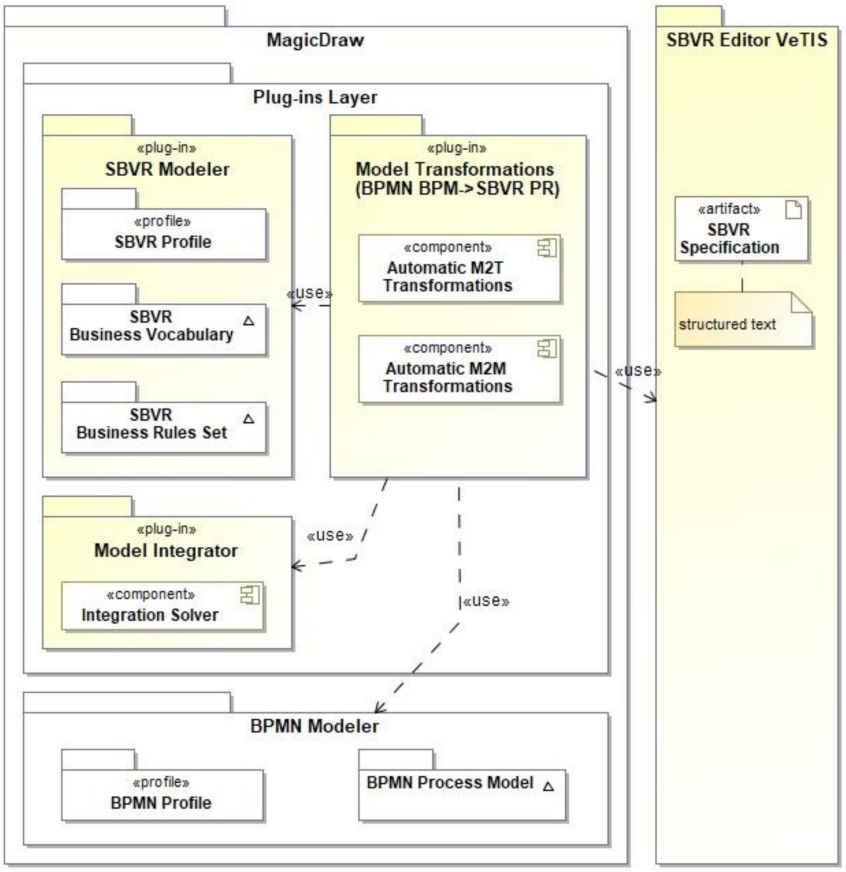


Figure 2. The basic architecture of the prototype of the model transformation solution.

To better understand the roles of each of the architectural components, let us take a look at the graphical interface of the overall solution (Figure 3):

- The input and output models (BPMN process model and SBVR business vocabulary and process rules, respectively) are displayed in the model view tab of the CASE tool (Tag 1). The presented models are stored as a single project representing the specified business knowledge of a business domain.
- Tag 2 denotes a toolbar comprising SBVR concepts for creating visual representations of SBVR business vocabularies (fact diagrams).
- In the center, there is the main area for deploying diagrams (Tag 3). The current view displays a fact diagram "Order Registration".
- On the right side of the CASE tool's graphical user interface, there is an additional viewer dedicated to exploring SBVR business vocabularies and rulesets (Tag 4).
- At the bottom, there is an optional frame opened to view a BPMN process diagram (Tag 5). Functionally, this frame is similar to the main frame representing the SBVR working environment. In MagicDraw, these two frames can switch places depending on the main focus of a user.
- The pop-up window (Tag 6) displays a condition of the opened BPMN process diagram, specified using so-called SBVR Structured English (SE).

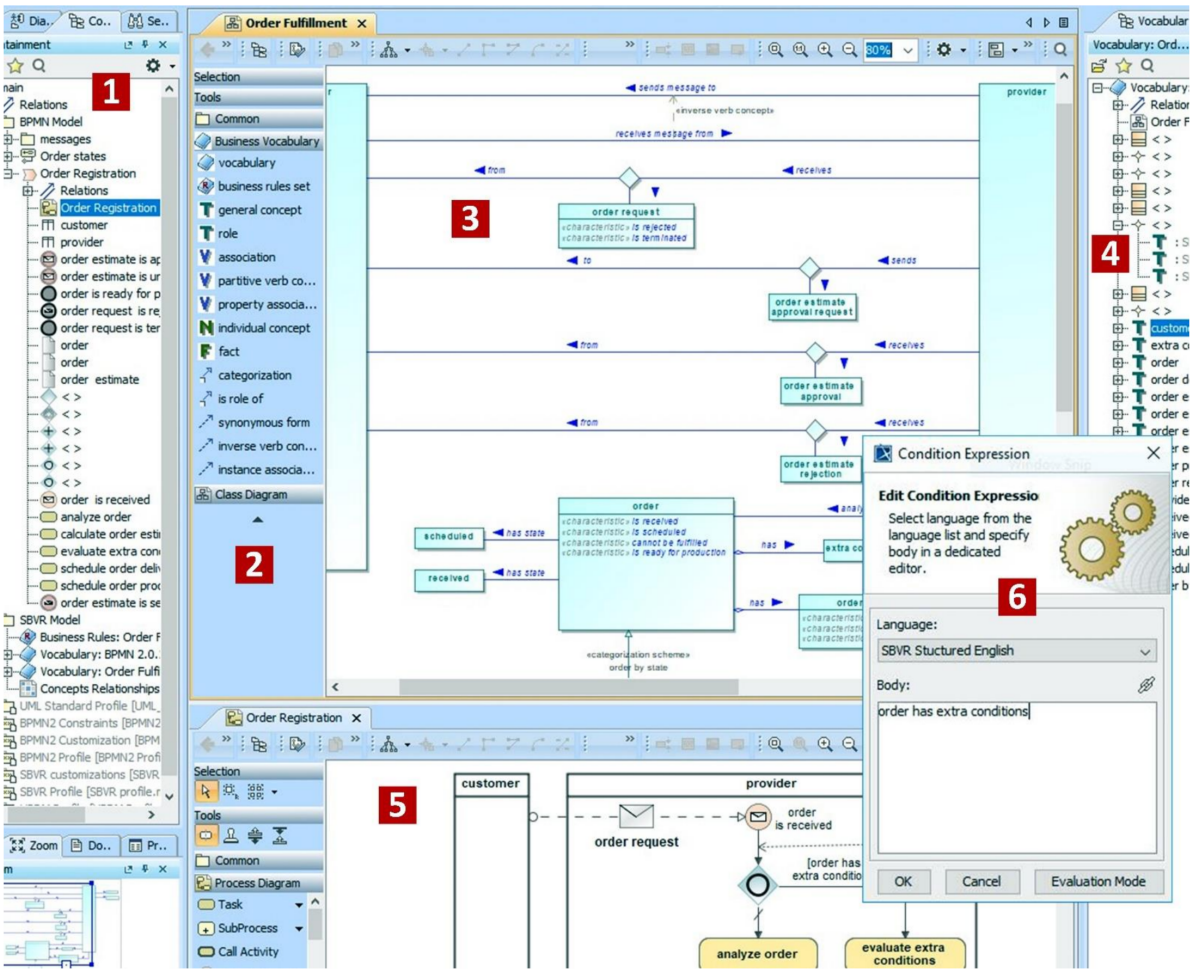


Figure 3. The MagicDraw GUI setup for working with BPMN and SBVR models.

Different sets of SBVR business rules (including process rules) can be managed through the customized MagicDraw tables (Figure 4).

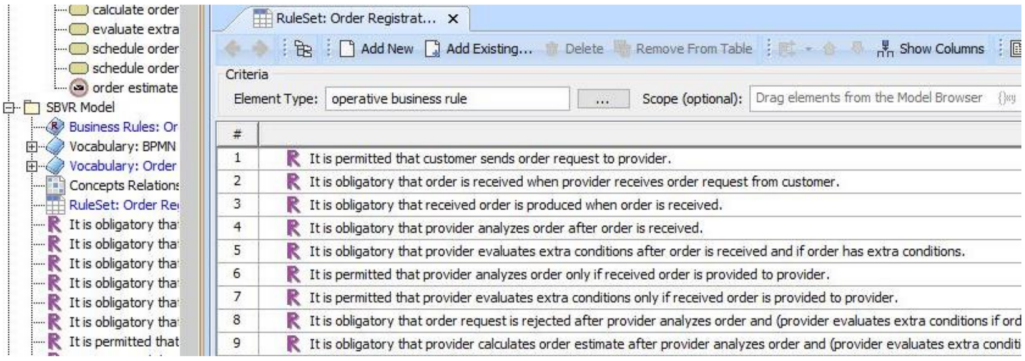


Figure 4. MagicDraw table for representing SBVR business rules (including process rules).

5. Illustrative Example

The section presents an example of a business process represented by the BPMN process diagram (Figure 5), which is an input of our model transformation approach, together with the resulting SBVR model consisting of SBVR business vocabulary and a set of process rules supported by the provided business vocabulary (Tables 2 and 3, respectively).

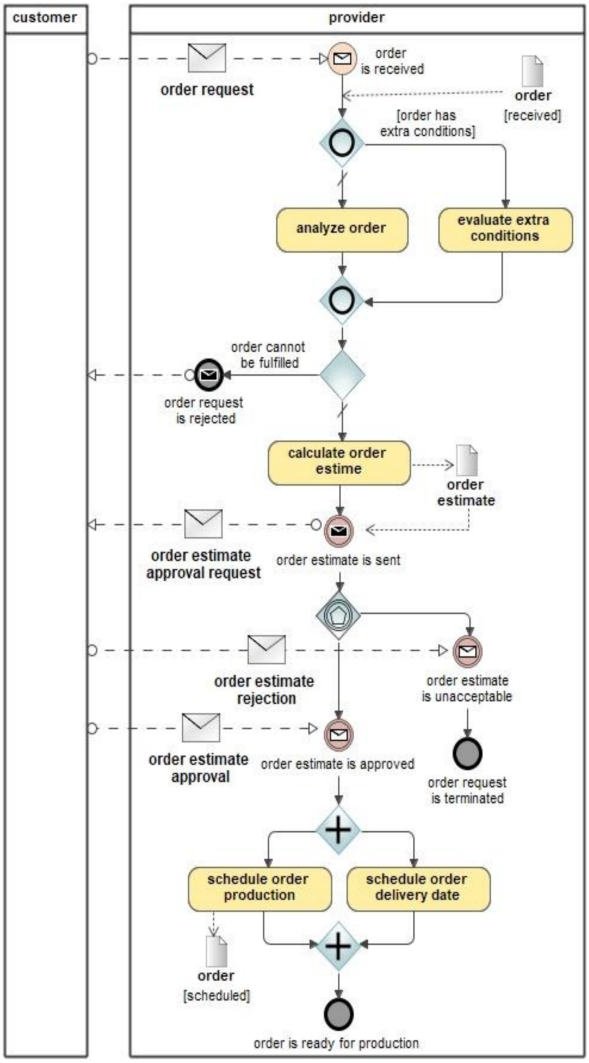


Figure 5. An illustrative example: the business process "Order Registration".

Table 2. An SBVR business vocabulary for the business process "Order Registration".

SBVR Concept Type	Extracted SBVR Concepts Representing an Exemplary Business Process
General concept	customer, provider, order, order request, received order, scheduled order, received, scheduled, order estimate, extra conditions, order estimate approval request, order estimate approval, order estimate rejection, order production, delivery date
Verb concept	customer *sends* order request *to* provider, provider *receives* order request *from* customer, order *is received*, order *has state* received, order *has* extra conditions, provider *analyzes* order, provider *evaluates* extra conditions, order *cannot be fulfilled*, order request *is rejected*, provider *sends message to* customer, customer *receives message from* provider, provider *calculates* order estimate, order estimate *is sent*, provider *sends* order estimate approval request *to* customer, customer *receives* order estimate approval request *from* provider, customer *sends* order estimate rejection *to* provider, provider *receives* order estimate rejection *from* customer, customer *sends* order estimate approval *to* provider, provider *receives* order estimate approval*from* customer, order estimate *is unacceptable*, order estimate *is approved*, order request *is terminated*, provider *schedules* order production, order *has state* scheduled, provider *schedules* order delivery date, order *is ready for production*

Once again, the extraction of SBVR business vocabularies from BPMN process models was not within the scope of this paper. However, general concepts and verb concepts form a basis for business rules (including process rules) and therefore they are also presented in this section (Table 2).

In Table 3, we organized the process rules into a sequence resembling the flow of work performed in the presented business process. This was done to increase the readability of

the specification. However, it must be noted that, by definition, SBVR business rules are not organized in any sequences within their rule sets.

Table 3. A set of SBVR process rules extracted from the business process "Order Registration".

SBVR Process Rule	Process Rule Pattern
It is permitted that customer *sends* order request *to* provider. It is obligatory that order *is received* when provider *receives* order request *from* customer.	T_9
It is obligatory that received order *is produced* when order *is received*.	T_7
It is obligatory that provider *analyzes* order after order *is received*. It is obligatory that provider *evaluates* extra conditionsafter order *is received* and if order *has* extra conditions.	T_3
It is permitted that provider *analyzes* order only if received order *is provided to* provider. It is permitted that provider *evaluates* extra conditionsonly if received order *is provided to* provider.	T_7
It is obligatory that order request *is rejected*after provider *analyzes* order and (provider *evaluates* extra conditionsif order *has* extra conditions) and if order *cannot be fulfilled*. It is obligatory that provider *calculates* order estimateafter provider *analyzes* order and (provider *evaluates* extra conditionsif order *has* extra conditions) and if not(order *cannot be fulfilled*).	***Cascading gateways***
It is obligatory that provider *sends message to* customerwhen order request *is rejected*. It is permitted that customer *receives message from* provider.	T_9
It is obligatory that order estimate *is produced* when provider *calculates* order estimate.	T_7
It is obligatory that order estimate *is sent* after provider *calculates* order estimate.	T_1
It is permitted that order estimate *is sent* only if order estimate *is provided to* provider.	T_7
It is obligatory that provider *sends* order estimate approval request*to* customerwhen order estimate *is sent*. It is obligatory that customer *receives* order estimate approval request*from* provider.	T_9
It is permitted that order estimate *is approved* after order estimate *is sent*. It is permitted that order estimate *is unacceptable* after order estimate *is sent*.	T_4
It is permitted that customer *sends* order estimate rejection*to* provider. It is obligatory that order estimate *is unacceptable* when provider *receives* order estimate rejection*from* customer.	T_9
It is permitted that customer *sends* order estimate approval*to* provider. It is obligatory that order estimate *is approved* when provider *receives* order estimate aproval*from* customer.	T_9
It is obligatory that order request *is terminated* after order estimate *is unacceptable*.	T_1
It is obligatory that provider *schedules* order production after order estimate *is approved*. It is obligatory that provider *schedules* order delivery dateafter order estimate *is approved*.	T_5
It is obligatory that scheduled order*is produced* when provider *schedules* order production.	T_7
It is obligatory that order *is ready for production*after provider *schedules* order production and provider *schedules* order delivery date.	T_5

6. Experimental Evaluation

For the experiment, our main goals were to verify whether: (1) the acquired process rules fully cover the workflows of the source BPMN process models; (2) our developed engineering solution fully complied with the conceptually defined approach for the extraction of SBVR process rules from BPMN process models.

Experiment settings. Thirty-two representative BPMN 2.0 process models of various sizes and complexity were selected from various internet sources for the experiment; the experimental transformation code together with the sets of source models, and the acquired results are available at: https://github.com/paudan/bpmn2sbvr (accessed on 29 August 2022). With the selected models, we tried to cover a variety of BPMN concepts relevant to our research as well as various situations of the flow of work within modeled business processes. After the models were selected, they were manually transferred to MagicDraw's modeling environment. At the same time, some models became subject to minor refactoring to eliminate the identified modeling and element naming issues. The resulting set of models was considered as an input for the experiment.

Experiment execution and results. At first, the experimental set of BPMN process models was manually processed by each of the authors of this research independently to acquire three sets of results comprising SBVR process rules extracted from the process

models using the defined set of transformation rules and the cascading gateways' algorithm. The acquired sets were then carefully compared to detect any conflicting situations. As natural language-related preprocessing is outside the scope of this paper, grammatical inconsistencies or ambiguities were not considered during the evaluation of the obtained results. Then, a single benchmark set of process rules was produced. After that, the automatic process rules extraction from the provided source BPMN models using the developed solution was performed to acquire automatic extraction results (Table 4).

Table 4. Statistics of the automatically extracted process rules.

Model	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	Cascading gtw.
Model 1	5	3	0	0	2	2	0	0	0	0
Model 2	7	4	0	0	0	0	0	0	10	0
Model 3	2	1	0	0	1	0	0	0	4	2
Model 4	2	1	0	0	2	0	0	0	4	2
Model 5	5	1	0	0	2	0	0	0	4	2
Model 6	2	2	2	0	3	0	0	0	0	1
Model 7	11	0	0	4	2	0	0	0	12	0
Model 8	4	3	0	0	0	0	0	0	0	0
Model 9	19	8	0	0	0	0	0	0	24	0
Model 10	4	2	0	0	0	0	0	0	12	0
Model 11	16	1	0	4	1	0	0	0	0	2
Model 12	4	2	0	0	3	0	0	0	0	0
Model 13	5	0	0	0	0	0	0	0	0	0
Model 14	5	0	0	4	0	0	0	0	0	0
Model 15	6	1	0	0	0	1	0	0	0	2
Model 16	5	2	0	0	3	0	0	0	0	0
Model 17	6	2	0	0	0	0	2	0	0	0
Model 18	15	6	0	0	0	0	5	0	12	0
Model 19	12	6	0	0	0	0	0	0	8	0
Model 20	8	4	0	0	0	0	0	0	16	0
Model 21	5	4	0	0	0	0	0	0	0	0
Model 22	14	6	0	0	0	0	3	0	10	0
Model 23	9	2	0	3	0	2	0	0	0	0
Model 24	7	6	0	0	0	0	4	0	0	0
Model 25	9	3	0	0	0	2	4	0	16	5
Model 26	7	2	0	2	0	0	0	0	10	0
Model 27	4	2	0	0	0	0	1	0	8	0
Model 28	7	2	0	0	3	0	4	0	6	0
Model 29	8	0	0	0	0	0	0	0	0	0
Model 30	4	2	0	0	0	0	0	0	6	0
Model 31	20	2	0	0	0	0	2	7	10	0
Model 32	6	2	0	0	0	0	4	0	2	0

In Table 4, the column *Model* denotes the 32 BPMN process models that were selected for the experiment; columns T_1–T_9 represent the transformation rules that were defined in

Table 1; each cell on the intersection of a specific source model *Model(i)* and transformation rule T_j represents the number of BPMN process rules that were acquired from *Model(i)* after performing T_j; the last column of Table 4 represents the results of the extraction of cascading gateways described in Section 3.3.

Discussion. The first observation from the acquired results of the automatic extraction of process rules (Table 4) is that, indeed, the experiment did cover a full set of the defined transformation rules and algorithms, meaning we have tested the whole specification. After closely analyzing the results, the main conclusion was that those results fully coincided with the benchmark results acquired from the manual extraction. Such a conclusion was drawn after comparing the process rules from the benchmark and automatic transformation result sets for each of the 32 source models. It should be reminded though that some of the source models were refactored, as described in the Experiment settings, to objectively avoid invalid entries. Previously, our early experimenting with the original, and therefore sometimes invalid, models was causing a certain number of invalid or even absent result entries. Registering and presenting such results in this paper was considered irrelevant as they were objectively influenced by the invalid source models and not by our development itself. However, such a situation confirmed the general observation from our other relevant model transformation research activities that modelers should put considerable effort into following best modeling practices and avoiding over-complex non-compliant models, thereby mitigating the possibility of different model interpretations, poor readability, and other related issues.

Yet another important observation was that the results covered the entirety of the specified flow of work defined in the experimental process models, meaning the set of specified transformation rules and algorithms was sufficient for the given scope of the experiment.

In addition, several other observations could be considered when talking about other possible applications of our development:

- An empirical investigation of the acquired sets of process rules showed that those rulesets could provide enough business knowledge to reverse engineer the flows of the initial processes with very little expert assistance, provided the process rules were supported by the underlying business vocabulary enriched with the BPMN concept types. A more in-depth investigation of this subject, however, does not fall within the scope of this paper.
- Assuming that the process model is one of the main sources of business knowledge for functional requirements elicitation, one could also consider utilizing the developed SBVR models for the same purpose. In such cases, the SBVR business vocabularies and rules could be used as an alternative, or at least a complementing, textual specification next to business process models and other sources of knowledge. Such an approach would fall in line with the already existing developments that use SBVR as one of the means to express requirements (e.g., [62]).
- While some still prefer UML activity diagrams to BPMN process diagrams when modeling business processes, the earlier-discussed motivation behind using the complementing natural language-based SBVR specifications remains the same. Therefore, it would be worth exploring the possibilities of adopting our developed approach for UML as well. We assume that such an adoption would utilize a subset of modified model transformation rules from our current development and the basic model transformation scenario (Figure 1) would remain unchanged.

Finally, it is important to state the limitations that are inherent in this approach:

- The processes should use pool/lane elements to define the participants. If this condition is failed, it will not be possible to extract subjects as noun concepts, which take part in the formation of verb concepts and, subsequentially, business rules.
- While we did not consider the semantic and syntactic validity of the labels (in other words, names) of model elements in this research, not naming the elements at all would result in the formation of invalid process rules.

- While there exist several conventions about naming conditions on exclusive and inclusive gateways, the specificity of the formation of SBVR verb concepts requires writing explicit condition formulations directly on the sequence flows leaving a gateway. Technical remark: the conditions can be written in the *Name*, or the *Guard* property of a sequence flow as shown in the exemplary process diagram in Figure 5.

In our experiment, all the above-mentioned limitations (or peculiarities, as we would like to call them) were handled during the refactoring of the selected source models.

Additionally, it should be noted that our development utilized only a subset of BPMN concepts that are relevant to the formation of process rules. This constraint was explicitly stated in Section 3.2.

7. Threats to Validity

Basic threats to the validity of our experiment results and the approach itself are categorized as follows [63]: (a) threats to construct validity, (b) threats to internal validity, and (c) threats to external validity. All three categories of threats are assessed in the following subsections.

Threats to Construct Validity. To properly apply any approach, one must ensure a common understanding of things among all interested parties. However, the interpretability and common understanding of concepts throughout the problem domain have the potential to be handled improperly in certain settings; this is called a threat to construct validity. This threat was minimal in our research, because: (1) all the specified transformation rules and algorithms were automated and did not leave any space for misinterpreting things; (2) the manual extraction of benchmark experimental results was performed exclusively by the authors, who naturally shared a common understanding of things.

Threats to Internal Validity. Internal validity considers internal factors and how they can affect the results, e.g., human error, motivation, subjectivity. Even though the automated nature of the experiment helped to reduce the influence of the human factor, some threats to internal validity remained.

The manual selection of source models for the experiment might be considered one of such threats because it holds a certain degree of subjectivity. Nonetheless, the models were carefully selected by considering the variety of possible modeling cases to test various capabilities of the approach. Therefore, we state that, in this case, the subjectivity in the selection of source models was justified.

Another threat to internal validity, which is a human error, is also related to the set of source models. As the BPMN process models were selected from different sources, neither of them was ready to be processed automatically by our experimental system. Human errors could appear during this preparation phase. However, all the inputs for the experiment were double-checked and agreed upon by all authors; therefore, we assume this threat to be minimal.

Threats to External Validity. External validity can be biased by the non-rigorous generalization or failure to generalize the developed approach to a larger extent. In our case, the main threat to external validity is the fact that a relatively small number of models was selected for the experiment; this objectively reduces the reliability of the acquired experiment results to some extent.

Another threat is the lack of standardized modeling practices, in particular, using certain conventions of naming model elements. In addition, there is always a chance that some practitioners will disagree on certain modeling practices used in this research, e.g., placing fully formulated conditions on the sequence flows diverging from a gateway instead of stating a question on the gateway itself and then placing "yes/no"-like answers on the diverging sequence flows. All this could lead to worse results compared with the ones presented in this paper.

8. Conclusions

Among other concerns that the professionals working in model-driven ISD face are the lack of efficient standards-based approaches that would provide practical and efficient means to the development of business process models synchronized with formalized well-structured business vocabularies and business rules specifications.

In this paper, basic conceptual and implementation aspects of the approach for the automatic extraction of SBVR process rules from BPMN process models were presented. At the core of the approach are nine transformation patterns and the two conceptual algorithms for processing cascading gateways. The experimental findings confirmed that SBVR process rules can indeed be extracted from BPMN business process models by both manual and automatic means. Moreover, the results of the automatic extraction of process rules have shown full compliance with the benchmark results, providing a solid background for the practical application and future developments of the solution. Another important observation was that the results covered the entirety of the specified flow of work defined in the experimental process models, meaning the set of specified transformation rules and algorithms was sufficient for the given scope of the experiment.

It is evident from the experimental evaluation that the quality of the results (especially in the case of automatic extraction) depends heavily on the quality of the underlying business vocabulary and whether the source models meet commonly accepted modeling practices.

The operating environment of our developed solution is a specific CASE tool, namely MagicDraw. However, the conceptually defined transformation patterns and the algorithm for processing cascading gateways could be implemented in any modeling tool supporting BPMN and SBVR standards (e.g., through UML extension mechanism) and third-party model transformations.

Future work. One of the most certain directions for our future research is the integration of the process rules transformation method with the advancements of NLP, in which we have already reached certain tangible results [62]. Such an enhancement would mitigate the NLP-related limitations of the existing transformation method. Another research effort should be focused on conceptually merging SBVR process vocabulary and process rule models with SBVR business vocabulary and business rules, thus gaining the ability to specify and represent static as well as dynamic aspects of business domain knowledge using the SBVR standard. Lastly, a thorough investigation of reverse-engineering textual SBVR process rules back to visual BPMN process models could be yet another relevant undertaking.

Author Contributions: Conceptualization, T.S.; Data curation, P.D., E.M. and R.B.; Methodology, T.S.; Resources, P.D.; Software, P.D.; Supervision, T.S.; Validation, T.S., P.D., E.M. and R.B.; Writing – original draft, T.S., P.D., E.M. and R.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Leopold, H.; Mendling, J.; Polyvyanyy, A. Generating Natural Language Texts from Business Process Models. In *Advanced Information Systems Engineering, LNCS, Proceedings of the 24th International Conference, CAiSE 2012, Gdansk, Poland, 25–29 June 2012*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7328, pp. 64–79. [CrossRef]
2. Wipp, W. Natural Language-Based Visualization and Modeling for Updatable Process Views. Bachelor's Thesis, Faculty of Engineering and Computer Science Institute of Databases and Information Systems, Ulm University, Ulm, Germany, 2013.
3. Leopold, H.; Mendling, J.; Polyvyanyy, A. Supporting process model validation through natural language generation. *IEEE Trans. Softw. Eng.* **2014**, *40*, 818–840. [CrossRef]
4. Rodrigues, R.A.; Azevedom, L.G.; Revoredo, K. BPMN2TEXT: A Language-Independent Framework to Generate Natural Language Text from BPMN models. iSys-Braz. *J. Inf. Syst.* **2016**, *9*, 38–56. [CrossRef]

5. Ottensooser, A.; Fekete, A.; Reijers, H.A.; Mendling, J.; Menictas, C. Making sense of business process descriptions: An experimental comparison of graphical and textual notations. *J. Syst. Softw.* **2012**, *85*, 596–606. [CrossRef]
6. Figl, K.; Recker, J. Exploring cognitive style and task-specific preferences for process representations. *Requir. Eng.* **2016**, *21*, 63–85. [CrossRef]
7. Object Management Group. *MDA Guide Revision 2.0*; OMG Document Number: Ormsc/14-06-01; Object Management Group: Needham, MA, USA, 2014.
8. Object Management Group. *Semantics of Business Vocabulary and Business Rules (SBVR) Specification, version 1.4*; OMG Document Number: Formal/17-05-05; Object Management Group: Needham, MA, USA, 2017.
9. Object Management Group. *Business Process Model and Notation (BPMN), version 2.0.1*; OMG Document Number: Formal/2013-09-02; Object Management Group: Needham, MA, USA, 2013.
10. Krogstie, J.; McBrien, P.; Owens, R.; Seltveit, A.H. Information systems development using a combination of process and rule based approaches. In *Advanced Information Systems Engineering, LNCS, Proceedings of the Third International Conference CAiSE'91, Trondheim, Norway, 13–15 May 1991*; Springer: Berlin/Heidelberg, Germany, 1991; Volume 498, pp. 319–335. [CrossRef]
11. Habich, D.; Richly, S.; Demuth, B.; Gietl, F.; Spilke, J.; Lehner, W.; Assman, U. Joining Business Rules and Business Processes. In Proceedings of the 16th International Conference on Information and Software Technologies (ICIST), Kaunas, Lithuania, 21–23 April 2010; pp. 361–368.
12. Sapkota, B.; van Sinderen, M. Exploiting rules and processes for increasing flexibility in service composition. In Proceedings of the 14th IEEE International Conference on Enterprise Distributed Object Computing Conference Workshops (EDOCW), Vitoria, Brazil, 25–29 October 2010; pp. 177–185. [CrossRef]
13. Koehler, J. The Process-Rule Continuum—Can BPMN & SBVR Cope with the Challenge? In Proceedings of the 2011 IEEE 13th Conference on Commerce and Enterprise Computing, Luxembourg-Kirchberg, Luxembourg, 5–7 September 2011; pp. 302–309. [CrossRef]
14. Milanovic, M.; Gaševic, D.; Rocha, L. Modeling Flexible Business Process with Business Rule Patterns. In Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Conference, Helsinki, Finland, 29 August–2 September 2011; pp. 65–74. [CrossRef]
15. Nalepa, G.J.; Kluza, K.; Kaczor, K. Proposal of an Inference Engine Architecture for Business Rules and Processes. In *Artificial Intelligence and Soft Computing, LNCS, Proceedings of the 12th International Conference, ICAISC 2013, Zakopane, Poland, 9–13 June 2013*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 453–464. [CrossRef]
16. Wang, W.; Indulska, M.; Sadiq, S. A Theoretical Perspective on Integrated Modeling of Business Processes and Rules. In Proceedings of the 28th International Conference on Advanced Information Systems Engineering (CAiSE Forum), Ljubljana, Slovenia, 13–17 June 2016; Springer: Ljubljana, Slovenia, 2016.
17. Kluza, K.; Nalepa, G.J. A Method for Generation and Design of Business Processes with Business Rules. *Inf. Softw. Technol.* **2017**, *91*, 123–141. [CrossRef]
18. Kluza, K.; Nalepa, G.J. Formal Model of Business Processes Integrated with Business Rules. *Inf. Syst. Front.* **2018**, *21*, 1167–1185. [CrossRef]
19. Rosa, L.S.; Silva, T.S.; Fantinato, M.; Thom, L.H. A visual approach for identification and annotation of business process elements in process descriptions. *Comput. Stand. Interfaces* **2022**, *81*, 103601. [CrossRef]
20. Aysolmaz, B.; Leopold, H.; Reijers, H.A.; Demirörs, O. A semi-automated approach for generating natural language requirements documents based on business process models. *Inf. Softw. Technol.* **2018**, *93*, 14–29. [CrossRef]
21. Vanhatalo, J.; Volzer, H.; Leymann, F. Faster and More Focused Control-Flow Analysis for Business Process Models Through SESE Decomposition. In *Service-Oriented Computing—ICSOC 2007, LNCS, Proceedings of the Fifth International Conference, Vienna, Austria, 17–20 September 2007*; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4749, pp. 43–55. [CrossRef]
22. Goncalves, J.C.; Santoro, F.M.; Baiao, F.A. Let Me Tell You a Story—On How to Build Process Models. *J. Univers. Comput. Sci.* **2011**, *17*, 276–295. [CrossRef]
23. Friedrich, F.; Mendling, J.; Puhlmann, F. Process Model Generation from Natural Language Text. In *Advanced Information Systems Engineering, LNCS, Proceedings of the 23rd International Conference, CAiSE 2011, London, UK, 20–24 June 2011*; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6741, pp. 482–496. [CrossRef]
24. Vakulenko, S. Extraction of Process Models from Business Process Descriptions. Master's Thesis, University of Tartu, Tartu, Estonia, 2011.
25. Van der Aa, H.; Carmona, J.; Leopold, H.; Mendling, J.; Padro, L. Challenges and Opportunities of Applying Natural Language Processing in Business Process Management. In Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, NM, USA, 20–26 August 2018; pp. 2791–2801. Available online: http://aclweb.org/anthology/C18-1236 (accessed on 10 July 2022).
26. Spreeuwenberg, S. Interview with Rob van Haarst, Author of the Recently-published Book "SBVR Made Easy". *Bus. Rules J.* **2014**, *15*. Available online: http://www.brcommunity.com/a2014/b747.html (accessed on 10 July 2022).
27. Nijssen, S. SBVR & BPMN as Pillars of Business Engineering. *Bus. Rules J.* 2008, 9. Available online: http://www.brcommunity.com/a2008/b447.html (accessed on 10 July 2022).
28. Zur Muehlen, M.; Indulska, M. Modeling Languages for Business Processes and Business Rules: A Representational Analysis. *Inf. Syst. J.* **2010**, *35*, 379–390. [CrossRef]

29. Eder, R.; Filieri, A.; Kurz, T.; Heistracher, T.J.; Pezzuto, M. Model-transformation-based software Generation Utilizing Natural Language Notations. In Proceedings of the 2nd IEEE International Conference on Digital Ecosystems and Technologies (DEST), Phitsanuloke, Thailand, 26–29 February 2008; pp. 306–312. [CrossRef]
30. Moschoyiannis, S.; Marinos, A.; Krause, P.J. Generating SQL Queries from SBVR Rules. In *Semantic Web Rules, Proceedings of the International Symposium, RuleML 2010, Washington, DC, USA, 21–23 October 2010*; Springer: Berlin/Heidelberg, Germany, 2010. [CrossRef]
31. Karpovic, J.; Ablonskis, L.; Nemuraitė, L.; Paradauskas, B. Experimental investigation of transformations from SBVR business vocabularies and business rules to OWL 2 ontologies. *Inf. Technol. Control* **2016**, *45*, 195–207. [CrossRef]
32. Essebaa, I.; Chantit, S. Tool Support to Automate Transformations from SBVR to UML Use Case Diagram. In Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE), Funchal, Portugal, 23–24 March 2018; pp. 525–532. [CrossRef]
33. Agrawal, A. Semantics of Business Process Vocabulary and Process Rules. In Proceedings of the 4th India Software Engineering Conference (ISEC), Kerala, India, 24–27 February 2011; pp. 61–68. [CrossRef]
34. Cheng, R.; Sadiq, S.; Indulska, M. Framework for Business Process and Rule Integration: A Case of BPMN and SBVR. In *Business Information Systems, LNBIP, Proceedings of the14th International Conference, BIS 2011, Poznań, Poland, 15–17 June 2011*; Springer: Berlin/Heidelberg, Germany, 2011; Volume 87, pp. 13–24. [CrossRef]
35. Skersys, T.; Tutkute, L.; Butleris, R.; Butkiene, R. Extending BPMN Business Process Model with SBVR Business Vocabulary and Rules. *Inf. Technol. Control* **2012**, *41*, 356–367. [CrossRef]
36. Skersys, T.; Kapocius, K.; Butleris, R.; Danikauskas, T. Extracting Business Vocabularies from Business Process Models: SBVR and BPMN Standards-based Approach. *Comput. Sci. Inf. Syst.* **2014**, *11*, 1515–1535. [CrossRef]
37. Mickeviciute, E.; Butleris, R.; Gudas, S.; Karciauskas, E. Transforming BPMN 2.0 Business Process Model into SBVR Business Vocabulary and Rules. *Inf. Technol. Control* **2017**, *46*, 360–371. [CrossRef]
38. Malik, S.; Bajwa, I.S. A Rule Based Approach for Business Rule Generation from Business Process Models. In *Rules on the Web: Research and Applications, LNCS, Proceedings of the 6th International Symposium, RuleML 2012, Montpellier, France, 27–29 August 2012*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7438, pp. 92–99. [CrossRef]
39. Malik, S.; Bajwa, I.S. Back to Origin: Transformation of Business Process Models to Business Rules. In *Business Process Management Workshops, Proceedings of the BPM 2012 International Workshops, LNBIP, Tallinn, Estonia, 3 September 2012*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 132, pp. 611–622. [CrossRef]
40. Al-Ali, H.; Damiani, E.; Al-Qutayri, M.; Abu-Matar, M.; Mizouni, R. Translating BPMN to Business Rules. In *Data-Driven Process Discovery and Analysis, Proceedings of the 6th IFIP WG 2.6 International Symposium, SIMPDA 2016, Graz, Austria, 15–16 December 2016*; Springer: Cham, Switzerland, 2016; pp. 22–36.
41. Rachdi, A.; En-Nouaary, A.; Dahchour, M. Analysis of common business rules in BPMN process models using business rule language. In Proceedings of the 11th International Conference on Intelligent Systems: Theories and Applications (SITA), Mohammedia, Morocco, 19–20 October 2016; pp. 1–6. [CrossRef]
42. Steen, B.; Pires, L.F.; Iacob, M. Automatic generation of optimal business processes from business rules. In Proceedings of the 4th IEEE International Enterprise Distributed Object Computing Conference Workshops, Vitoria, Brazil, 25–29 October 2010; pp. 117–126.
43. Wu, Z.; Yao, S.; He, G.; Xue, G. Rules Oriented Business Process Modeling. In Proceedings of the Internet Technology and Applications (iTAP), Wuhan, China, 16–18 August 2011; pp. 1–4. [CrossRef]
44. Tantan, O.C.; Akoka, J. Automated transformation of Business Rules into Business Processes. In Proceedings of the Twenty-Sixth International Conference on Software Engineering and Knowledge Engineering, Vancouver, Canada, 1–3 July 2014; pp. 684–687.
45. Addamssiri, N.; Kriouile, A.; Boussaa, S.; Gadi, T. MDA Approach: Refinement and Validation of CIM Level Using SBVR. In Proceedings of the Mediterranean Conference on Information & Communication Technologies, Saïdia, Morocco, 7–9 May 2015. [CrossRef]
46. Kluza, K.; Honkisz, K. From SBVR to BPMN and DMN Models. Proposal of Translation from Rules to Process and Decision Models. In *Artificial Intelligence and Soft Computing, LNCS, Proceedings of the 5th International Conference, ICAISC 2016, Zakopane, Poland, 12–16 June 2016*; Springer: Cham, Switzerland, 2016; Volume 9693, p. 9693. [CrossRef]
47. Elkindy, A.I.A. Survey of Business Process Modeling Recommender Systems. Master's Thesis, Universitat Koblenz-Landau, Mainz, Germany, 2019. Available online: https://kola.opus.hbz-nrw.de/frontdoor/index/index/docId/1895 (accessed on 10 July 2022).
48. Sola, D.; van der Aa, H.; Meilicke, C.; Stuckenschmidt, H. Exploiting label semantics for rule-based activity recommendation in business process modeling. *Inf. Syst.* **2022**, *108*, 102049. [CrossRef]
49. Deng, S.; Wang, D.; Li, Y.; Cao, B.; Yin, J.; Wu, Z.; Zhou, M. A Recommendation System to Facilitate Business Process Modeling. *IEEE Trans. Cybern.* **2017**, *47*, 1380–1394. [CrossRef]
50. Gottesdiener, E. Business RULES Show Power, Promise. *Appl. Dev. Trends* **1977**, *4*, 36–42. Available online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.329.8585&rep=rep1&type=pdf (accessed on 10 July 2022).
51. Ross, R. *The Business Rule Book: Classifying, Defining and Modeling Rules*, 2nd ed.; Business Rule Solutions Inc.: Houston, TX, USA, 1997.

52. Business Rules Group. Defining Business Rules—What Are They Really? Final Report, Revision 1.3, July, 2000 (Formerly Known as the "GUIDE Business Rules Project Report," 1995). Available online: http://www.businessrulesgroup.org/first_paper/BRG-whatisBR_3ed.pdf (accessed on 10 July 2022).
53. Vanthienen, J.; Goedertier, S. How Business Rules Define Business Processes. *Bus. Rules J.* **2007**, *8*. Available online: http://www.brcommunity.com/a2007/b336.html (accessed on 10 July 2022).
54. Baisley, D.E. A Metamodel for Business Vocabulary and Rules: Object-Oriented Meets Fact-Oriented. *Bus. Rules J.* **2004**, *5*. Available online: http://www.brcommunity.com/a2004/b197.html (accessed on 10 July 2022).
55. Cabot, J.; Pau, R.; Raventos, R. From UML/OCL to SBVR specifications: A challenging transformation. *Inf. Syst.* **2010**, *35*, 417–440. [CrossRef]
56. Kleiner, M.; Albert, P.; Bezivin, J. Parsing SBVR-Based Controlled Languages. In *Model Driven Engineering Languages and Systems, Proceedings of the 12th International Conference, MODELS 2009, LNCS, Denver, CO, USA, 4–9 October 2009*; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5795, pp. 122–136. [CrossRef]
57. Business Rules Group (BRG). The Business Rules Manifesto. 2003. Available online: Businessrulesgroup.org/brmanifesto.htm (accessed on 10 July 2022).
58. Skersys, T.; Danenas, P.; Butleris, R. Extracting SBVR business vocabularies and business rules from UML use case diagrams. *J. Syst. Softw.* **2018**, *141*, 111–130. [CrossRef]
59. Czarnecki, K.; Helsen, S. Classification of Model Transformation Approaches. In Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture, Twente, The Netherlands, 26–27 June 2003; Volume 45, pp. 1–17.
60. Mens, T.; van Gorp, P. A Taxonomy of Model Transformation. *Electron. Notes Theor. Comput. Sci.* **2006**, *152*, 125–142. [CrossRef]
61. Object Management Group. *Query/View/Transformation (QVT), version 1.1*; OMG Document Number: Formal/2011-01-01; Object Management Group: Needham, MA, USA, 2011.
62. Danenas, P.; Skersys, T.; Butleris, R. Natural language processing-enhanced extraction of SBVR business vocabularies and business rules from UML use case diagrams. *Data Knowl. Eng.* **2020**, *128*, 101822. [CrossRef]
63. Runeson, P.; Höst, M.; Rainer, A.; Regnell, B. *Case Study Research in Software Engineering: Guidelines and Examples*; John Wiley & Sons: Hoboken, NJ, USA, 2012.