



Formal Model of Business Processes Integrated with Business Rules

Krzysztof Kluza¹ · Grzegorz J. Nalepa¹

Published online: 19 February 2018
© The Author(s) 2018. This article is an open access publication

Abstract

Business Processes describe the ways in which operations are carried out in order to accomplish the intended objectives of organizations. A process can be depicted using a modeling notation, such as Business Process Model and Notation. Process model can also describe operational aspects of every task; but in a properly designed model, the detailed aspects of low-level logic should be delegated to external services, especially to a Business Rule Engine. Business Rules support the specification of knowledge in a declarative manner and can be successfully used for specification of details for process tasks and events. However, there is no unified model of process integrated with rules that ensures data types consistency. Thus, the paper presents a formal description of the integration of Business Processes with Business Rules. We provide a general model for such integration as well as the model applied to a specific rule representation from the Semantic Knowledge Engineering approach.

Keywords Business process modeling · Business rules · Process with rules integration

1 Introduction

Business Process Management (BPM) (Dumas et al. 2013; Weske 2012) is a holistic approach for improving organization's workflow in order to align processes with client needs. It focuses on reengineering of processes to obtain optimization of procedures, increase efficiency and effectiveness by the constant process improvement.

According to this approach, a Business Process (BP) can be simply defined as a collection of related tasks which produces a specific service or product for a customer (Lindsay et al. 2003). Models of BPs are intended to be a bridge between technical and business people. They are simple and visualizations make them much easier to understand than using a textual description. Thus, modeling is an essential part of BPM.

Although processes provide a universal method of describing operational aspects of business, detailed aspects of process logic should be described on different abstraction level. Business Rules (BR) can be successfully used to specify process low-level logic (Charfi and Mezini 2004; Knolmayer et al. 2000). What is important, the BR approach supports the specification of knowledge in a declarative manner.

There is a difference in abstraction levels of BP and BR; however, rules can be to a certain degree complementary to processes. BR provide a declarative specification of domain knowledge, which can be encoded into a process model (Kluza et al. 2012). On the other hand, a process can be used as a procedural specification of the workflow, including the inference control.

The use of BR in BP design helps to simplify complex decision modeling. It was also empirically proven that the effect of linked rules improved business process model understanding (better time efficiency in interpreting business operations, less mental effort, improved accuracy of understanding) (Wang et al. 2017). Although rules should describe business knowledge in a formalized way that can be further automated (Nalepa and Ligęza 2005), there is no common understanding how process and rule models should be structured in order to be integrated (Hohwiller et al. 2011). There is also no formalized model that combines processes with rules and ensures data types consistency, i.e. data types used in rules should be available in processes that use these rules. This is the main challenge we wish to address in our work.

The research presented in this paper is supported by the HiBuProBuRul Project funded from NCN (National Science Centre) resources for science (no. DEC-2011/03/N/ST6/00909).

✉ Krzysztof Kluza
kluza@agh.edu.pl

Grzegorz J. Nalepa
gjn@agh.edu.pl

¹ AGH University of Science and Technology,
al. A. Mickiewicza 30, 30-059 Krakow, Poland

As a solution, to the above mentioned challenge, we propose a model for describing integration of BP with BR. Our starting point are some existing representation methods for processes and rules, specifically Business Process Model and Notation (BPMN) (OMG 2011a) for BP models, and eXtended Tabular Trees version 2 (XTT2) (Nalepa et al. 2011c) which is a formalized rule representation developed as a part of the Semantic Knowledge Engineering (SKE) approach (Nalepa 2011). We extend these models, and on top of them provide a formalized general business logic model that incorporates rules in the process models. The model uses the abstract rule representation, so it can be refined by adjusting it to the specific rule representation. In the paper, we use XTT2 for this purpose, as it is a fully formalized rule representation. In our case, the proposed model deals with the integration of processes with rules to provide a coherent formal description of the integrated model.

Usually, the goal of formal model description is its formal verification. In our case, the proposed model deals with the integration of processes with rules to provide a coherent formal description of the integrated model. Although we do not consider the formal verification of such a model, as we take advantage of the existing representations, the formal verification of each part of the integrated model (process or rules) is possible. Because of using the XTT2 rule representation, it is possible to use the existing methods for formal verification of rules (Ligeza 1999), especially for formal verification of decision components, such as inconsistency, completeness, subsumption or equivalence (Nalepa et al. 2011a). In the case of process model itself, one can use the existing verification methods (Wynn et al. 2009), specifically the methods that can take into account the task logic (Szpyrka et al. 2011).

The paper is organized as follows. Section 2 presents an overview of related works concerning process model formalization. Section 3 provides a formal description of a BPMN process model. It introduces the notation and its formal representation. This formalized process model is then integrated with rules, and this integration is specified as the General Business Logic Model in Section 4. In order to apply this model to a specific rule solution, the SKE-specific Business Logic Model is presented in Section 5. As an evaluation, a case example described using the proposed model is presented in Section 6. The paper is concluded in Section 7.

2 Related Works

There are several attempts to the formalization of Business Process models. These formalizations differ with respect to the goal with which the model semantics is specified.

2.1 Formalization of Business Process Models

Ouyang et al. (2006a, b) introduced a formal description of BPMN process model for the purpose of their translation to Business Process Execution Language (BPEL), in order to execute process models. They also defined the concept of well-structured components for the purpose of the execution. Dijkman et al. (2007) defined the formal semantics of BPMN process model in order to use formal analysis. The main goal of their formalization was to define the execution semantics to enable soundness checking. In Dijkman and Gorp (2011), they formalized execution semantics of BPMN through translation to Graph Rewrite Rules. Such formalization can support simulation, animation and execution of BPMN 2.0 models. It can help in conformance testing of implementations of the workflow engines that use the notation for modeling and executing processes.

In 2002, Sivaraman and Kamath (2002) used Petri nets for modeling Business Processes. However, it was before the BPMN notation was introduced. In fact, the translation from BPMN to Petri nets is not straightforward. Changizi et al. (2010) formalized BP model through translation to channel-based coordination language Reo. Such transformations allows for verification of models with the help of verification and model checking tools available for the Reo language, such as mCRL2 model checker. Speck et al. (2011) formalized Event-driven Process Chain (EPC) diagrams using Computational Tree Logic (CTL). With CTL, it is possible to formulate rules about different paths in processes and use that for checking the existence of a specific element type in the process, for unknown elements or elements with only partially known properties. Wong and Gibbons (2008, 2011) defined BPMN model semantics in terms of the Z model for syntax and Communicating Sequential Processes (CSP) for behavioral semantics. This allows for checking the consistency of models at different levels of abstraction as well as other properties that must apply to the process, such as domain specific properties, deadlock-freeness or proper completion.

Lam (2009, 2012) formally defined the token-based semantics of BPMN models in terms of Linear Temporal Logic (LTL). This allows for verification and reasoning on BPMN models, especially for checking such properties as liveness or reachability. Ligeza (2011) defined a declarative model for well-defined BPMN diagrams, which allows for specification of correct components and correct dataflow, especially checking model consistency or termination condition. Smirnov et al. (2012) introduced simple formalization for defining action patterns in business process model repositories. Action patterns capture different relations among actions often observed within process model collections. The formalization used in Smirnov et al. (2012)

allowed the authors extract various types of action patterns from industrial process model collections.

Bădică et al. (2003) presented the method for modeling business processes using Role Activity Diagrams which share many similarities with BPMN. They used the formalization that exploits Finite State Process Algebra (FSP) which is suitable for verification of these models using Fluent Linear Temporal Logic (FLTL) (Bădică and Bădică 2011).

2.2 Decision Model and Notation and Business Process Models

On the other hand, there is a new standard for modeling decisions of organizations – Decision Model and Notation (DMN) (OMG 2015). The DMN specification shows examples how to link decisions in business process with specific elements in the process model. However, it does not specify the details of this relationship. Moreover, it emphasizes the independence of the DMN of the business process and BPMN. The handbook for process and decision modeling (Debevoise et al. 2014) presents how to separate decision logic to DMN during modeling, but it does not specify the formal relationships between the models and their integration issues.

Janssens et al. (2016) specified five integration scenarios of decision and processes, which covers the continuum from processes without decisions (scenario 0) to decisions without processes (scenario 4). Our approach can be located somewhere between scenario 1, in which local decisions ensure separation of control flow and decision logic, and scenario 2, where decisions are placed directly before the business activities requiring their output. This paper systematize the integration possibilities, but it does not consider the data integration between models.

Batoulis et al. (2015b) proposed a method for extracting decision logic from process models. They specified three patterns for control-flow-based decision identification and provide an algorithm for transforming such decision logic to DMN model. This is suitable for refactoring process models in order to separate decision logic from process models. Bazhenova and Weske (2015) proposed an approach for deriving decision models from process models and execution logs. The approach consists of four steps: decision points identification, decision logic finding, constructing decision model and adapting process model. Thus, it requires the event log of a process model. Another approach which dynamically adapts the decision logic based on historic and current process execution data in BPM system during run-time was proposed in Batoulis et al. (2015a). In this approach, the decisions are automatically revealed and translated to DMN what improves process executions regarding such issues as process duration or

costs. Catalkaya et al. (2013) proposed the method for applying decision theory to process models and extend the current standard with rule-based XOR-split gateway (the so called r-gateway). In order to take a decision the method utilize key performance indicators, such as cost of performing an activity.

It is important to emphasize that DMN relation to BPMN is described in the “Annex A: Relation to BPMN” of the DMN specification (OMG 2015) only in the informative way. Linking BPMN and DMN Models is described informally. Thus, no constraints or requirements for data are imposed.

The formalizations mentioned above were used either for formal analysis of the model (checking selected properties) or its execution. In the case of the DMN representation, there are research directions providing the way of deriving decision models from process models and describing the possibility of execution of such models. However, in the approach presented in this paper, we focus on designing the integrated models, so the purpose of the model is to formally describe the integration of the BP model with rules and to provide the basis for the formal description of other integration issues. It is partially based on the BPMN formalization proposed by Ouyang et al. (2006a). Yet we extend this formalization by incorporating rules into process models, which is our original contribution.

3 Formal Description of BPMN Process Model

Let us define a BPMN 2.0 process model that describes the most important artifacts of the BPMN notation.¹ As it focuses also on several details that are key elements from the rule perspective, the process model takes into account flow objects (activities, events and gateways), sequence flows between these flow objects, as well as the set of model attributes.

Definition 1 A BPMN 2.0 process model is a tuple $\mathcal{P} = (\mathcal{O}, \mathcal{F}, \Lambda)$, where:

- \mathcal{O} is the set of flow objects, $o_1, o_2, o_3, \dots \in \mathcal{O}$,
- Λ is the set of model attributes, $\lambda_1, \lambda_2, \lambda_3, \dots \in \Lambda$,
- $\mathcal{F} \subset \mathcal{O} \times \mathcal{O} \times 2^{\Lambda_{\mathcal{F}}}$ is the set of sequence flows, where $\Lambda_{\mathcal{F}} \subset \Lambda$ is a subset of attributes that are used in sequence flows.

Moreover, the set of flow objects is divided into sets:

- \mathcal{A} is the set of activities such that $\mathcal{A} = \mathcal{T} \cup \mathcal{S}$, $\mathcal{T} \cap \mathcal{S} = \emptyset$, where \mathcal{T} is the set of tasks and \mathcal{S} is the set of

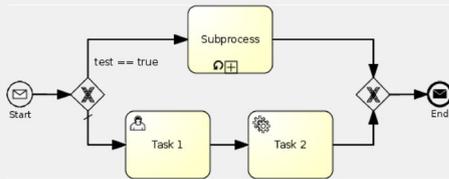
¹The model is limited to the most popular elements used in the private process model diagrams of the BPMN 2.0 notation.

- sub-processes, $\tau_1, \tau_2, \tau_3, \dots \in \mathcal{T}$ and $s_1, s_2, s_3, \dots \in \mathcal{S}$,
- \mathcal{E} is the set of events, $e_1, e_2, e_3, \dots \in \mathcal{E}$
- \mathcal{G} is the set of gateways, $g_1, g_2, g_3, \dots \in \mathcal{G}$

such that $\mathcal{O} = \mathcal{A} \cup \mathcal{E} \cup \mathcal{G}$ and $\mathcal{A} \cap \mathcal{E} = \mathcal{A} \cap \mathcal{G} = \mathcal{E} \cap \mathcal{G} = \emptyset$.

The set of all the possible BPMN 2.0 process models will be denoted \mathcal{P} , e.g. $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \dots \in \mathcal{P}$.

Example 1 The process model shown below presents an exemplary BPMN model consisting of two events, two gateways and two tasks.



Using the presented formalization, it can be described as follows: $\mathcal{P}_1 = (\mathcal{O}_1, \mathcal{F}_1, \Lambda_1)$, where

- $\mathcal{O}_1 = \{\tau_1, \tau_2, s_1, e_1, e_2, g_1, g_2\}$, and \mathcal{O}_1 can be divided into following subsets:
- $\mathcal{O}_1 = \mathcal{A}_1 \cup \mathcal{E}_1 \cup \mathcal{G}_1, \mathcal{A}_1 = \mathcal{T}_1 \cup \mathcal{S}_1, \mathcal{T}_1 = \{\tau_1, \tau_2\}$ and $\mathcal{S}_1 = \{s_1\}, \mathcal{E}_1 = \{e_1, e_2\}, \mathcal{G}_1 = \{g_1, g_2\}$,
- $\mathcal{F}_1 = \{(e_1, g_1, \Lambda_{e_1, g_1}), (g_1, s_1, \Lambda_{g_1, s_1}), (g_1, \tau_1, \Lambda_{g_1, \tau_1}), (\tau_1, \tau_2, \Lambda_{\tau_1, \tau_2}), (s_1, g_2, \Lambda_{s_1, g_2}), (\tau_2, g_2, \Lambda_{\tau_2, g_2}), (g_2, e_2), \Lambda_{g_2, e_2}\}$, where $\Lambda_{e_1, g_1}, \Lambda_{g_1, s_1}, \Lambda_{g_1, \tau_1}, \Lambda_{\tau_1, \tau_1}, \Lambda_{s_1, g_2}, \Lambda_{\tau_2, g_2}, \Lambda_{g_2, e_2} \subset \Lambda_1$ are sets of sequence flows' attributes.

Definition 2 A task interpretation is a pair:

$$\tau = (type(\tau), \Lambda_\tau)$$

where:

- $type(\tau)$ determines the type of the task τ , $type(\tau) \in \{None, User, Manual, Send, Receive, Script, Service, Business Rule\}$
- $\Lambda_\tau \subset \Lambda$ is the set of the task attributes, $\Lambda_\tau = \{id, name, documentation, markers, resources, \dots, implementation\}$,² some attributes can take set values, such as:

- $markers \subset \{loop, parallelMI, sequentialMI, adhoc, compensation\}$,

some of the attributes may contain other attributes, such as:

- $ioSpecification = \{dataInputs, dataOutputs\}$.

²The set of attributes is limited to the most popular ones, for other see Section 10.2 Activities of the BPMN 2.0 Specification (OMG 2011a).

Moreover, \mathcal{T}_x will denote the set of tasks of the same type x :

$$\mathcal{T}_x = \{\tau \in \mathcal{T} : type(\tau) = x\}$$

For simplicity, the value of an attribute can be obtained using a function, name of which matches the attribute name $attribute(\tau)$, e.g. $id(\tau_1)$ denotes the value of the *id* attribute for the task τ_1 .

The tasks of different type use the *implementation* attribute to specify the implementation technology, e.g. “##WebService” for the Web service technology or a URI identifying any other technology or coordination protocol. The purpose of the implementation technology is different for different types of tasks, e.g.: in the case of Service tasks ($\mathcal{T}_{Service}$) it determines the service technology, in the case of Send tasks (\mathcal{T}_{Send}) or Receive tasks ($\mathcal{T}_{Receive}$), it specifies the technology that will be used to send or receive messages respectively, and in the case of Business Rule tasks ($\mathcal{T}_{Business Rule}$), it specifies the Business Rules Engine technology. The *ioSpecification* attribute is used for specifying the data inputs and outputs suitable for the implementation technology. Some types of tasks can also have several additional attributes (Λ_τ) specified.

Example 2 For the process model presented in Example 1, tasks can be described as follows:

- $\mathcal{T}_1 = \mathcal{T}_{User} \cup \mathcal{T}_{Service}, \mathcal{T}_{User} = \{\tau_1\}, \mathcal{T}_{Service} = \{\tau_2\}$,
- $name(\tau_1) = \text{“Task 1”}, name(\tau_2) = \text{“Task 2”}, markers(\tau_1) = markers(\tau_2) = \emptyset$.

Definition 3 A sub-process interpretation is a triple:

$$s = (\mathcal{P}_s, type(s), \Lambda_s)$$

where:

- $type(s)$ determines the type of the sub-process s , $type(s) \in \{Sub-process, Embedded, CallActivity, Transaction, Event\}$,
- $\mathcal{P}_s \in \mathcal{P}$ is a BPMN 2.0 process model nested in the sub-process s ,
- $\Lambda_s \subset \Lambda$ is the set of the sub-process attributes, $\Lambda_s = \{id, name, documentation, markers, \dots, triggeredByEvent\}$,³ such attributes are defined same as for tasks (see Definition 2).

³The set of attributes is limited to the most popular ones, for other see Section 10.2 Activities of the BPMN 2.0 Specification (OMG 2011a).

Example 3 For the process model presented in Example 1, a sub-process $s_1 \in \mathcal{S}_I$ can be described as follows:

- $name(s_1) = \text{“Subprocess”}$,
- $type(s_1) = \text{Sub-process}$,
- $markers(s_1) = \{loop\}$,
- $triggeredByEvent(s_1) = \text{false}$,
- $loopCharacteristics(s_1) = \{loopCondition(s_1), testBefore(s_1), loopMaximum(s_1)\}$, where:

- $loopCondition(s_1) \neq \text{null}$,
- $testBefore(s_1) = \text{false}$,
- $loopMaximum(s_1) = \text{null}$,

where null indicates that the value does not exist.

While activities represent parts of work that are performed within a Business Process, events are state changes occurring in the process environment that can influence the process.

Definition 4 An event interpretation is a pair:

$$e = (type(e), trigger(e), \Lambda_e)$$

where:

- $type(e) \in \{Start, Intermediate, End\}$,
- $trigger(e)$ determines the trigger of the event e , $trigger(e) \in \{Cancel, Compensation, Conditional, Error, Escalation, Link, Message, Multiple, None, ParallelMultiple, Signal, Terminate, Timer\}$,
- $\Lambda_e \subset \Lambda$ is the set of the event attributes, $\Lambda_e = \{id, name, documentation, method, boundary, attachedToRef, cancelActivity\}$,⁴ $method(e) \in \{catch, throw\}$.

Moreover, \mathcal{E}^x will denote the set of events with the same trigger x :

$$\mathcal{E}^x = \{e \in \mathcal{E} : trigger(e) = x\}$$

\mathcal{E}_x will denote the set of events of the same type x :

$$\mathcal{E}_x = \{e \in \mathcal{E} : type(e) = x\}$$

Different types of events have different event definition attributes specified, e.g.:

- *messageEventDefinition* for $e \in \mathcal{E}^{Message}$, $messageEventDefinition = \{messageRef, operationRef\}$,

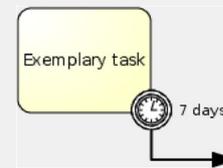
⁴The set of attributes is limited to the most popular ones, for other see Section 10.4 Events of the BPMN 2.0 Specification (OMG 2011a).

- *timerEventDefinition* for $e \in \mathcal{E}^{Timer}$, $timerEventDefinition = \{timeCycle, timeDate, timeDuration\}$,
- *conditionalEventDefinition* for $e \in \mathcal{E}^{Conditional}$, $conditionalEventDefinition = \{condition\}$,

Example 4 For the process model presented in Example 1, the events $e_1, e_2 \in \mathcal{E}_I$ can be described as follows:

- $name(e_1) = \text{“Start”}$, $name(e_2) = \text{“End”}$,
- $method(e_1) = \text{catch}$, $method(e_2) = \text{throw}$,
- $e_1 \in \mathcal{E}_{Start}^{Message}$ and $e_2 \in \mathcal{E}_{End}^{Message}$,
- $type(e_1) = \text{Start}$, $type(e_2) = \text{End}$,
- $trigger(e_1) = \text{Message}$, $trigger(e_2) = \text{Message}$,
- $boundary(e_1) = \text{boundary}(e_2) = \text{false}$,
- $cancelActivity(e_1) = \text{cancelActivity}(e_2) = \text{false}$.

Example 5 For the fragment of a process model with a task τ_3 and an event e_3 presented below:



the following aspects can be described using such a model:

- $e_3 \in \mathcal{E}_{Intermediate}^{Timer}$,
- $name(e_3) = \text{“7 days”}$,
- $method(e_1) = \text{catch}$
- $timeDuration(e_3) = \text{“P7D”}$,^a
- $timeCycle(e_3) = \text{timeDate}(e_3) = \text{null}$,
- $cancelActivity(e_3) = \text{true}$,
- $boundary(e_3) = \text{true}$,
- $attachedToRef(e_3) = id(\tau_3)$.

^aThe format used for time duration specification is the ISO 8601 format (as required by the BPMN 2.0 specification).

It is also important to note that not every $trigger(e)$ is allowed for any $type(e)$ of event – Table 1 presents the possible combinations.

In the case of this formalization, the *condition* attribute for $e \in \mathcal{E}^{Conditional}$ is especially important. It defines an expression stored in *body* attribute and expressed in the *language* language: $condition(e) = \{body(e), language(e)\}$.

Gateway elements are used to control the flow of tokens through sequence flows as they converge and diverge within a process. Although, according to the BPMN 2.0 specification, a single gateway can have multiple input and multiple output flows, this formalized model proposed is to

Table 1 Possible combinations of $trigger(e)$ and $type(e)$

$type(e)$	$Start$	$Intermediate$		End
	$method(e) = catch$	$method(e) = catch$	$method(e) = throw$	$method(e) = throw$
<i>Cancel</i>		✓		✓
<i>Compensation</i>	✓	✓		✓
<i>Conditional</i>	✓	✓		
<i>Error</i>	✓	✓		✓
<i>Escalation</i>	✓	✓		✓
<i>Link</i>		✓	✓	
<i>Message</i>	✓	✓	✓	✓
<i>Multiple</i>	✓	✓	✓	✓
<i>None</i>	✓		✓	✓
<i>ParallelMultiple</i>	✓	✓		
<i>Signal</i>	✓	✓	✓	✓
<i>Terminate</i>				✓
<i>Timer</i>	✓	✓		

enforce a best practice of a gateway only performing one of these functions. Thus, a gateway should have either one input or one output flow and a gateway with multiple input and output flows should be modeled with two sequential gateways, the first of which converges and the second diverges the sequence flows.

Definition 5 A gateway interpretation is a tuple:

$$g = (\mathcal{F}_g^{in}, \mathcal{F}_g^{out}, type(g), \Lambda_g)$$

where:

- \mathcal{F}_g^{in} and \mathcal{F}_g^{out} are sets of sequence flows (input and output flows respectively), $\mathcal{F}_g^{in}, \mathcal{F}_g^{out} \subset \mathcal{F}$, $\mathcal{F}_g^{in} = \{(o_i, o_j, \Lambda_{i,j}) \in \mathcal{F} : o_j = g\}$ and $\mathcal{F}_g^{out} = \{(o_i, o_j, \Lambda_{i,j}) \in \mathcal{F} : o_i = g\}$,
- $type(g)$ determines the type of the gateway g , $type(g) \in \{Parallel, Exclusive, Inclusive, Complex, Event-based, ParallelEvent-based\}$,
- $\Lambda_g \subset \Lambda$ is the set of the gateway attributes, $\Lambda_g = \{id, name, documentation, gatewayDirection\}$,⁵ $gatewayDirection(g) \in \{converging, diverging\}$.

Furthermore, the following notational elements will be used:

- $\mathcal{G}_+ = \{g \in \mathcal{G} : type(g) = Parallel\}$,
- $\mathcal{G}_\times = \{g \in \mathcal{G} : type(g) = Exclusive\}$,
- $\mathcal{G}_\circ = \{g \in \mathcal{G} : type(g) = Inclusive\}$,
- $\mathcal{G}_* = \{g \in \mathcal{G} : type(g) = Complex\}$,

⁵The set of attributes is limited to the most popular ones, for other, see Section 10.5 Gateways of the BPMN 2.0 Specification (OMG 2011a).

- $\mathcal{G}_\otimes = \{g \in \mathcal{G} : type(g) = Event-based\}$,
- $\mathcal{G}_\oplus = \{g \in \mathcal{G} : type(g) = ParallelEvent-based\}$.

Some types of gateways can have several additional attributes specified, such as:

- $\{instantiate, eventGatewayType\}$ for $g \in \mathcal{G}_\otimes \cup \mathcal{G}_\oplus$, $instantiate(g) \in \{true, false\}$, $eventGatewayType(g) \in \{Parallel, Exclusive\}$,
- $\{default\}$ for $g \in \mathcal{G}_\times \cup \mathcal{G}_\circ \cup \mathcal{G}_*$, $default(g) \in \mathcal{F}_{out} \cup \{null\}$.

Example 6 For the process model presented in Example 1, the gateways $g_1, g_2 \in \mathcal{G}_I$ can be described as follows:

- $type(g_1) = type(g_2) = Exclusive$, thus $g_1, g_2 \in \mathcal{G}_\times$,
- $gatewayDirection(g_1) = diverging$, $gatewayDirection(g_2) = converging$.
- $\mathcal{F}_{g_1}^{in} = \{(e_1, g_1, \Lambda_{e_1, g_1})\}$ and $\mathcal{F}_{g_1}^{out} = \{(g_1, s_1, \Lambda_{g_1, s_1}), (g_1, \tau_1, \Lambda_{g_1, \tau_1})\}$,
- $\mathcal{F}_{g_2}^{in} = \{(s_1, g_2, \Lambda_{s_1, g_2}), (\tau_2, g_2, \Lambda_{\tau_2, g_2})\}$ and $\mathcal{F}_{g_2}^{out} = \{(g_2, e_2, \Lambda_{g_2, e_2})\}$,
- $default(g_1) = (g_1, \tau_1, \Lambda_{g_1, \tau_1})$, $default(g_1) = null$,

Sequence Flows are used for connecting flow objects $o \in \mathcal{O}$ in the process.

Definition 6 A sequence flow interpretation is a tuple:

$$f_{o_1, o_2} = ((o_1, o_2), \Lambda_{o_1, o_2})$$

where:

- $(o_1, o_2) \in \mathcal{O} \times \mathcal{O}$ and o_1, o_2 are respectively source and target elements,
- $\Lambda_{o_1, o_2} \subset \Lambda_{\mathcal{F}}$ is the set of sequence flow attributes, $\Lambda_{o_1, o_2} = \{id, name, documentation, default, conditional, condition\}$,⁶ $condition = \{body, language\}$.

Two boolean attributes: *conditional* and *default* determine the conditional or default type of the flow. A conditional flow has to specify the *condition* and a default flow has no condition, i.e.

- $conditional(f) = true \Rightarrow condition(f) \neq null$,
- $default(f) = true \Rightarrow condition(f) = null$.

A subset of conditional sequence flows will be denoted $\mathcal{F}_{Conditional}$, i.e. $\mathcal{F}_{Conditional} = \{f \in \mathcal{F} : conditional(f) = true\}$,

A *condition* attribute defines an expression indicating that the token will be passed down the sequence flow only if the expression evaluates to `true`. An expression *body* is basically specified using natural-language text. However, it can be interpreted as a formal expression by a process execution engine; in such case, BPMN provides an additional *language* attribute that specifies a language in which the logic of the expression is captured.

In the presented BPMN model, the evaluation of the value can be obtained using a function $eval(value)$, e.g. for the *condition* attribute of the f sequence flow: $eval(condition(f)) \in \{true, false\}$. If *condition* is not explicitly defined for a particular sequence flow f , then it is implicitly always evaluated to `true`, i.e.: $condition(f) = null \Rightarrow eval(condition(f)) \equiv true$.

Example 7 For the process model presented in Example 1, the output sequence flows from gateway g_1 , i.e. $f \in \mathcal{F}_{g_1}^{out}$, can be described as follows:

- $f_{g_1, s_1} = (g_1, s_1, \Lambda_{g_1, s_1})$ and $f_{g_1, \tau_1} = (g_1, \tau_1, \Lambda_{g_1, \tau_1})$.
- $default(f_{g_1, s_1}) = false$ and $default(f_{g_1, \tau_1}) = true$,
- $conditional(f_{g_1, s_1}) = true$ and $conditional(f_{g_1, \tau_1}) = false$,
- $condition(f_{g_1, s_1}) = \{body(f_{g_1, s_1}), language(f_{g_1, s_1})\}$, $body(f_{g_1, s_1}) = "test == true"$,
- $condition(f_{g_1, \tau_1}) = null$.

⁶The set of attributes is limited to the most popular ones, for other, see Section 10.5 Gateways of the BPMN 2.0 Specification (OMG 2011a).

In this section, we presented a formalized model of BPMN business process. This model will be used in the following section for defining a model that combines business processes with business rules.

4 General Business Logic Model

In this section we define a General Business Logic Model, which specifies business logic as the knowledge stored in the form of processes integrated with rules. The model follows the ideas from the BPMN formalization proposed by Ouyang et al. (2006a). However, we extend the process model by incorporating rules into process models. Such integrated models are called business logic models. As the model uses the abstract rule representation, it is general and can be refined into a specific one by adjusting it to the specific rule representation.

The model uses the process model presented in the previous section and integrates it with rules.⁷ As rules constitute a part of a rule base, it is defined as follows.

Definition 7 A **Rule Base** is a tuple $\mathbb{K} = (\mathbb{A}, \mathbb{R}, \mathbb{T})$, where:

- \mathbb{A} is the set of all attributes used in the rule base,
- \mathbb{R} is the set of all rules, $r_1, r_2, r_3, \dots \in \mathbb{R}$, and a single rule r_i contains its conditional part denoted as $cond(r_i)$.⁸
- \mathbb{T} is the set of all decision components, $t_1, t_2, t_3, \dots \in \mathbb{T}$, which can be rule sets or more sophisticated structures (rule sets represented as decision tables, trees, etc.) that organize rules in the rule base ($\mathbb{T} \subset 2^{\mathbb{R}}$).

Moreover, it is assumed that each rule base specifies a *language(r)*⁹ in which the rules are specified and provides additional methods that can be used to obtain pieces of information from the rule base, such as $eval(r)$ ¹⁰ for evaluating a conditional part of the rule, and $infer(t)$ ¹¹ for obtaining a result of inference on a specified rule set.

Definition 8 A **General Business Logic Model** is a tuple $\mathcal{M} = (\mathcal{P}, \mathbb{K}, map)$, where:

- \mathbb{K} is a rule base containing rules (as defined in Definition 7),

⁷The proposed model focuses on selected details that are important from the rule perspective.

⁸It is assumed that various rule bases can contain different kinds of rules (see categories of rules presented in Wagner et al. 2005). Regardless of the kind, in every rule it is possible to isolate their conditional part. In some cases, a rule may consist only of a conditional part.

⁹ $language(r)$ denotes $language(\mathbb{K})$.

¹⁰ $eval(r)$ denotes $eval(cond(r))$ and $eval(r) \in \{true, false\}$.

¹¹ $infer(r)$ denotes $infer(\{r\})$

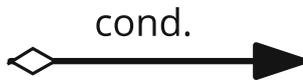


Fig. 1 Conditional sequence flow

- \mathcal{P} is a BPMN 2.0 process model (as defined in Definition 1),
- map is a map function defined as follows:

$$map(x) = \begin{cases} \mathcal{F}_{Conditional} \rightarrow \mathbb{R} & \text{for } x \in \mathcal{F}_{Conditional} \\ \mathcal{E}_{Conditional} \rightarrow \mathbb{R} & \text{for } x \in \mathcal{E}_{Conditional} \\ \mathcal{T}_{BusinessRule} \rightarrow \mathbb{T} & \text{for } x \in \mathcal{T}_{BusinessRule} \end{cases}$$

In the following paragraphs, the mapping details for the specific BPMN elements and more complex BPMN constructs are presented.

Conditional Sequence Flow For a Conditional Sequence Flow $f \in \mathcal{F}_{Conditional}$ (see Fig. 1) the following requirements have to be fulfilled in \mathcal{M} :

- All BPMN conditional sequence flows in \mathcal{P} have the condition in the form of a conditional part of a rule from the \mathbb{K} rule base assigned, formalized, i.e. the following holds:

$$\forall f \in \mathcal{F}_{Conditional} \exists r \in \mathbb{R} (map(f) = r) \wedge (body(f) = cond(r)) \wedge (language(f) = language(r)).$$

- All condition attributes $\mathbb{A}_r \subset \mathbb{A}$ required by the rule r should be available in the \mathcal{P} model, i.e.:

$$\forall r \in \mathbb{R} ((\exists f \in \mathcal{F} map(f) = r) \Rightarrow (\forall \lambda \in cond(r) \lambda \in \Lambda_{\mathcal{F}})).$$

Conditional Event Conditional Event $e \in \mathcal{E}_{Conditional}$ denotes that a particular *condition* specified by a rule condition is fulfilled. For Conditional Event, the following requirements have to be fulfilled in \mathcal{M} :

- All BPMN conditional events in \mathcal{P} have the condition in the form of a conditional part of a rule from the \mathbb{K} rule base assigned, i.e.:

$$\forall e \in \mathcal{E}_{Conditional} \exists r \in \mathbb{R} (map(e) = r) \wedge (body(e) = cond(r)) \wedge (language(e) = language(r)).$$

- All condition attributes $\mathbb{A}_r \subset \mathbb{A}$ required by the rule r should be available in the \mathcal{P} model, i.e.:

$$\forall r \in \mathbb{R} ((\exists e \in \mathcal{E} map(e) = r) \Rightarrow (\forall \lambda \in cond(r) \lambda \in \Lambda_e)).$$

A Conditional Event can be used in BPMN in several structures in order to support different situations based on the evaluation of the condition expression in the process instance, such as:

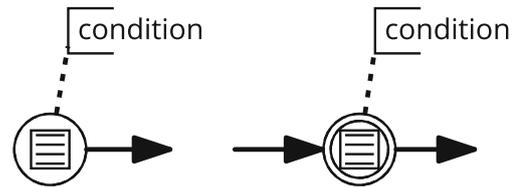


Fig. 2 Conditional (start and intermediate) events

- Simple Start and Intermediate Conditional Event can be used as conditional triggers providing the ability to trigger the flow of a token. The notation for conditional start and intermediate events are presented in Fig. 2.
- Non-interruptive and Interruptive Boundary Conditional Events attached to a Task or a Subprocess can be used for interrupting a task or subprocess. The notation for conditional non-interruptive and interruptive boundary events are presented in Fig. 3.
- Event Subprocess with Conditional Start Event can be used for interrupting the process and initiating a subprocess that is not a part of the regular control flow starting from the conditional start event. The notation for conditional non-interruptive and interruptive boundary events are presented in Fig. 4.

Business Rule Task Business Rule (BR) Tasks allow for specification of the task logic using rules and delegating work to a Business Rules Engine in order to receive calculated or inferred data. The notation for BR task is presented in Fig. 5.

For the BPMN Business Rule tasks, the following formulas have to be fulfilled in \mathcal{M} :

- All BPMN BR tasks in \mathcal{P} have the decision component from the \mathbb{K} rule base assigned, i.e.:

$$\forall \tau \in \mathcal{T}_{BusinessRule} \exists t \in \mathbb{T} map(\tau) = t.$$

- All the input attributes required by the Business Rules Engine for a rule set specified by the decision component should be available in the process model, i.e.:

$$\forall \tau \in \mathcal{T}_{BusinessRule} \forall r \in \mathbb{R} \forall \lambda \in cond(r) \lambda \in dataInputs(\tau). \\ t \in \mathbb{T} \\ map(\tau) = t$$

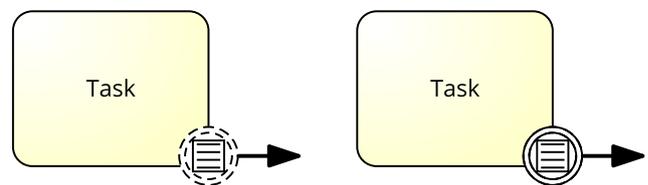


Fig. 3 Conditional (non-interruptive and interruptive) boundary events

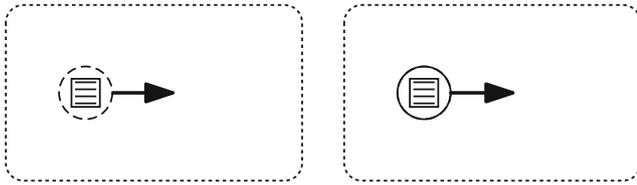


Fig. 4 Event subprocesses with conditional start event

- All the output attributes from the result of inference on a specified rule set from the Business Rules Engine should be available as the output of BR task in the process, i.e.:

$$\forall \tau \in \mathcal{T}_{BusinessRule} \quad \forall r \in \mathbb{R} \quad \forall \lambda \in infer(r) \quad \lambda \in dataOutputs(\tau) \\ t \in \mathbb{T} \\ map(\tau) = t$$

Diverging (Exclusive, Inclusive/Multi-choice and Complex) Gateways Gateways provide mechanisms for diverging a branch into two or more branches, and passing token from the incoming branch to one or more outgoing branches according to the type of a gateway.

For further formulae, the following sets are defined:

$$\mathcal{G}_{div}^{cond} = \{g \in \mathcal{G}_x \cup \mathcal{G}_o \cup \mathcal{G}_* : gatewayDirection(g) = diverging\}, \\ \mathcal{F}_{g,div}^{out,cond} = \{f \in \mathcal{F}_g^{out} : g \in \mathcal{G}_{div}^{cond} \wedge default(f) \neq true\}, \\ \mathcal{F}_{g,default}^{out,cond} = \{f \in \mathcal{F}_g^{out} : g \in \mathcal{G}_{div}^{cond} \wedge default(f) = true\}.$$

In the case of exclusive (\mathcal{G}_x), inclusive (\mathcal{G}_o) and complex (\mathcal{G}_*) diverging gateways (see Fig. 6), there is a need for the model \mathcal{M} to satisfy the following requirements:

- All BPMN sequence flows (apart from the default ones) outgoing from a diverging gateway have the condition in the form of a conditional part of a rule from the \mathbb{K} rule base assigned, i.e.:

$$\forall f \in \mathcal{F}_{g,div}^{out,cond} \quad \exists r \in \mathbb{R} \quad (map(f) = r) \wedge (body(f) = cond(r)) \\ \wedge (language(f) = language(r)).$$

- In the case of exclusive, inclusive and complex diverging gateways, they can have maximum one outgoing default sequence flow, i.e.:

$$\forall g \in \mathcal{G}_{div}^{cond} \quad |\mathcal{F}_{g,default}^{out,cond}| \leq 1.$$

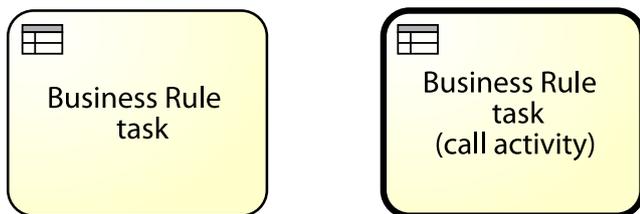


Fig. 5 Business rule task (a standard and a call activity task)

- In the case of exclusive gateways, the evaluated conditions have to be exclusive, i.e.:

$$\forall f_1, f_2 \in \mathcal{F}_{g,div}^{out} \quad \forall g \in \mathcal{G}_x \quad (\exists r_1, r_2 \in \mathbb{R} \quad map(f_1) = r_1 \wedge map(f_2) = r_2) \Rightarrow (eval(r_1) \neq eval(r_2)).$$

Converging Complex Gateway In the case of converging exclusive, inclusive and parallel gateways, their semantics is defined by the BPMN 2.0 specification and they do not require any rule-based description. However, a Converging Complex Gateway (see Fig. 7) requires an additional *activationCondition* expression which describes the precise behavior (defines the rule of passing tokens).

Thus, for BPMN Converging Complex Gateways the following requirements have to be fulfilled in \mathcal{M} :

- All BPMN Converging Complex Gateways in \mathcal{P} specify the rule of passing tokens, i.e.:

$$\forall g \in \mathcal{G}_* \quad \exists r \in \mathbb{R} \quad (map(g) = r) \wedge (body(g) = cond(r)) \\ \wedge (language(g) = language(r)),$$

where $activationCondition(g) = \{body(g), language(g)\}.$

- All condition attributes $\mathbb{A}_r \subset \mathbb{A}$ required by the rule r should be available in the process model, i.e.:

$$\forall r \in \mathbb{R} \quad ((\exists g \in \mathcal{G}_* \quad map(g) = r) \Rightarrow (\forall \lambda \in cond(r) \quad \lambda \in \Lambda_g)).$$

Gateway Preceded by a BR Task A special case of the two aboved examples occurs when a gateway is preceded by the BR task (Fig. 8). In the such case, there is a need for the model \mathcal{M} to satisfy the requirements specified for Business Rule Tasks and for Gateways, as well as the following additional requirement:

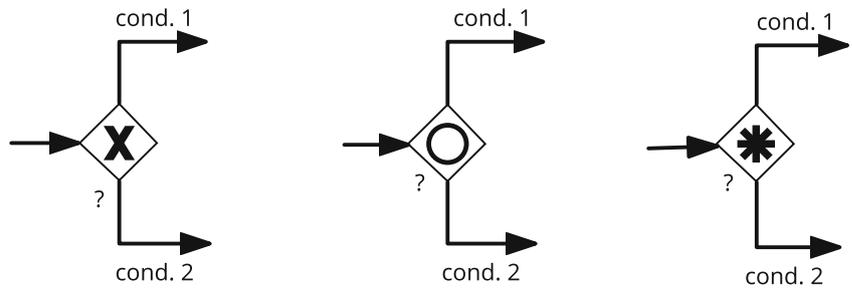
- All BPMN sequence flows (apart from the default sequence flows) outgoing from a diverging gateway preceded by the BR task have the conditions based on the output attributes of the BR task, i.e.:

$$\forall \tau \in \mathcal{T}_{BusinessRule} \quad \forall g \in \mathcal{G}_{div}^{cond} \quad ((\tau, g, \lambda_{\tau g}) \in \mathcal{F}) \\ t \in \mathbb{T} \\ map(\tau) = t \\ \Rightarrow \left(\forall f \in \mathcal{F}_g^{out} \quad \forall \lambda \in body(f) \quad \lambda \in infer(t) \right).$$

Gateway Preceded by a Subprocess Another special case of using a gateway is a gateway preceded by a subprocess in which a decision is made (see Fig. 9). In such the case, there is a need for the model \mathcal{M} to satisfy the requirements specified for Diverging Gateways, as well as the following additional requirements:

- All BPMN sequence flows (apart from the default sequence flows) outgoing from a diverging gateway

Fig. 6 Exclusive, inclusive (multi-choice) and complex diverging gateways



preceded by a subprocess have the conditions based on the attributes set by the preceded subprocess:

$$\forall s \in \mathcal{S} \forall g \in \mathcal{G}_{div}^{cond} ((s, g, \lambda_{s,g}) \in \mathcal{F}) \Rightarrow (\forall f \in \mathcal{F}_{out} \forall \lambda \in body(f) \lambda \in dataOutputs(s)).$$

- The number of sequence flows outgoing from a diverging gateway should be greater than or equal to the number of Message or None end events in the subprocess, i.e.:

Let: $\mathcal{E}_{End}^s = \{e \in \mathcal{O}_s : e \in \mathcal{E}_{End}^{None} \vee e \in \mathcal{E}_{End}^{Message}\}$, where $\mathcal{P}_s = (\mathcal{O}_s, \mathcal{F}_s, \Lambda_s)$.

$$\forall s \in \mathcal{S} \forall g \in \mathcal{G}_{div}^{cond} ((s, g, \lambda_{s,g}) \in \mathcal{F}) \Rightarrow (|\mathcal{F}_{out}^s| \geq |\mathcal{E}_{End}^s|).$$

Event-Based Gateway The use of Event-based (Exclusive) Gateway extends the use of Conditional Events (see Fig. 10). Thus, in this case, there is a need for the model \mathcal{M} to satisfy the requirements specified for Conditional Events, as well as the following additional requirements:

- All conditions in the Conditional Events that occur after the Event-based (Exclusive) Gateway should be exclusive,¹² i.e.:

$$\forall e_1, e_1 \in \mathcal{E}_{Conditional} \forall g \in \mathcal{G}_{\otimes} ((g, e_1, \lambda_{g,e_1}), r_1, r_1 \in \mathbb{R} \mapsto map(e_1) = r_1 \mapsto map(e_2) = r_2 (g, e_2, \lambda_{g,e_2}) \in \mathcal{F}) \Rightarrow \neg (eval(r_1) \wedge eval(r_2)).$$

Other BPMN Constructs Although other BPMN elements or constructs are not directly associated with rules from the rule base, they can be described by rules. However, such a representation of rules is not formally defined in the model presented here.

In this section, a simple model of the integration of the BP model with rules was proposed. Moreover, this formal description provides the basis for refinement of

the model for specific rule representation, e.g. the XTT2 representation from the SKE approach, which will be presented in the following sections.

5 SKE-Specific Business Logic Model

5.1 Semantic Knowledge Engineering Rules

Semantic Knowledge Engineering (SKE) (Nalepa 2011) is an approach that provides a coherent formal framework for eXtended Tabular Trees version 2 (XTT2) (Nalepa et al. 2011b), which is a rule-based knowledge representation language. The formalization of the XTT2 method based on the ALSV(FD) logic was presented by Nalepa and Ligęza in Nalepa (2010b, 2011). The goal of this section is to introduce the SKE-specific Business Logic Model, based on the General Business Logic Model presented in the previous section.

In order to formally define an XTT2 rule, the formal definitions for Attributive Logic with Set Values over Finite Domains (ALSV(FD)) are needed. ALSV(FD) is an extended version of Set Attributive Logic (SAL) (Ligęza 2006) oriented toward Finite Domains. It was discussed in (Nalepa 2010b). The expressive power of the ALSV(FD) formalism is increased through the introduction of new relational symbols. For simplicity, there are no objects specified in an explicit way. In ALSV(FD), A denotes the set of all attributes used to describe the system. Each attribute $a_i \in A, i = 1 \dots n$ has a set of admissible values that it takes (a domain) denoted as \mathbb{D}_i . Any domain is assumed to be a finite set.

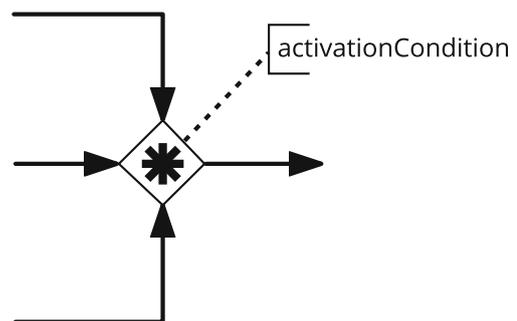
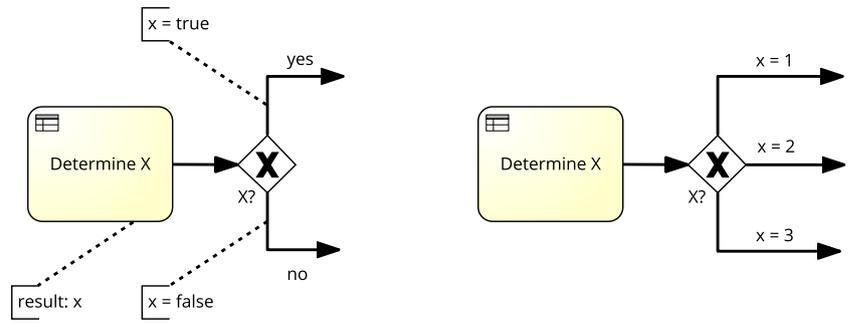


Fig. 7 Converging complex gateway

¹²In fact, the exclusive relation here applies only to evaluation to true values. Thus, both conditions can be not fulfilled at the same time.

Fig. 8 Gateway after BR task



Definition 9 An **ALSV(FD) triple** ε (also called a legal atomic formula in terms of the ALSV(FD) logic) is a triple of the form:

$$\varepsilon = (a_i, \Delta, x_i)$$

where

- a_i is an attribute,
- x_i is the value of the attribute (it can be a single element of the domain or a subset of the domain),
- Δ is an operator from the set of operators for attributes,¹³ such that $\Delta \in \{=, \neq, :=, \in, \notin, \subseteq, \supseteq, \sim, \not\sim\}$.

The ALSV(FD) triples constitute the basic components of the XTT2 rules. A set consisting of the ALSV(FD) triples will be denoted by E , i.e. $\varepsilon_1, \varepsilon_2, \dots \in E$.

Definition 10 An **XTT2 rule** r_i (here called also rule) is a triple of the form:

$$r_i = (cond_i, dec_i)$$

where

- $cond_i \subseteq E$ is a conditional part of a rule consisting of legal atomic formulae in terms of ALSV(FD),
- $dec_i \subseteq E$ is a decision part of a rule consisting of legal atomic formulae in terms of ALSV(FD).

The set of all rules will be denoted as R , and $r_1, r_2, \dots \in R$. A rule schema for a given rule r_i (called also rule template) is a pair $schema(r_i) = (A_i^{cond}, A_i^{dec})$, where A_i^{cond} and A_i^{dec} are sets of all the attributes occurring in the conditional and decision part of the rule respectively.

From a logical point of view, the order of the ALSV(FD) atomic formulae in both the conditional and decision parts of the rule is unimportant. Having the structure of a single rule defined, the structure of the XTT2 rule base can be defined. The rule base is composed of tables grouping rules having the same lists of attributes (rule schemas). Rule schemas are used to identify rules working in the same situation (operational context). Such a set of rules can

¹³The assignment operator ($:=$) allows for assigning a new value to an attribute.

form a decision component in the form of a decision table. A common schema can also be considered as a table header.

Definition 11 An **XTT2 decision component** t (also called an XTT2 table) is a sequence of rules having the same rule schema:

$$t = (r_1, r_2, \dots, r_n)$$

The set of decision components is denoted as T , $t_1, t_2, \dots \in T$. An XTT2 table schema (also called a schema of the component or table header) is denoted as $schema(t)$.

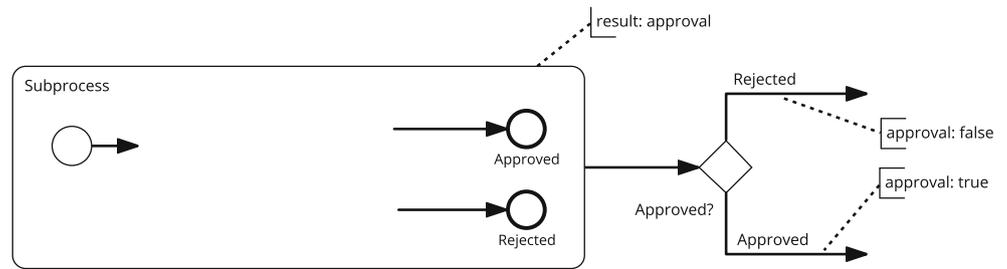
Example 8 An example of the decision component for the Business Rule task “Calculate base charge” with the following schema: $schema(t) = (\{carCapacity\}, \{baseCharge\})$ is presented in the below table. Such a component calculates the baseCharge price based on the carCapacity (the capacity of the car engine).

(?) carCapacity	(->) baseCharge
< 900	537.00
in [901, 1300]	753.00
in [1301, 1600]	1050.00
in [1601, 2000]	1338.00
>= 2000	1536.00

In the presented formalisms, this component can be represented as follows:

$$\begin{aligned}
 t &= (r_1, r_2, r_3, r_4, r_5), \\
 r_1 &= (\{(carCapacity, <, 900)\}, \\
 &\quad \{(baseCharge, :=, 537.00)\}), \\
 r_2 &= (\{(carCapacity, in, [901, 1300])\}, \\
 &\quad \{(baseCharge, :=, 753.00)\}), \\
 r_3 &= (\{(carCapacity, in, [1301, 1600])\}, \\
 &\quad \{(baseCharge, :=, 1050.00)\}), \\
 r_4 &= (\{(carCapacity, in, [1601, 2000])\}, \\
 &\quad \{(baseCharge, :=, 1338.00)\}), \\
 r_5 &= (\{(carCapacity, >=, 2000)\}, \\
 &\quad \{(baseCharge, :=, 1536.00)\}).
 \end{aligned}$$

Fig. 9 Gateway preceded by a subprocess



5.2 Integrated Model

The SKE-specific Business Logic Model is a special case of the General Business Logic Model that describes the integration of the BPMN process models with the SKE rules.

Definition 12 SKE-specific Business Logic Model is a tuple: $\mathcal{M}_{SKE} = (\mathcal{P}, \mathbb{K}_{SKE}, map)$, where:

- $\mathbb{K}_{SKE} = (\mathbb{A}_{SKE}, \mathbb{R}_{SKE}, \mathbb{T}_{SKE}) = (A, R, T_{\mathcal{X}})$ is an SKE-specific rule base, where:

- $T_{\mathcal{X}}$ is a set of the XTT2 decision components,
- R is a set of the XTT2 rules, such as:

$$R = \{r_i \in t : t \in T_{\mathcal{X}}\},$$

$$\forall r_i \in R \text{ schema}(r_i) = \text{schema}(t),$$

and the conditional $cond(r_i)$ part of a rule is defined as follows:

$$cond(r_i) = E_i^{cond},$$

where $r_i = (E_i^{cond}, E_i^{dec}, ACT_i)$,

- A is a set of the attributes used in the XTT2 rule base, i.e.:¹⁴

$$A = \{a_i : \exists r_i \in R a_i \in A_i^{cond} \vee a_i \in A_i^{dec}\}.$$

- \mathcal{P} is a BPMN 2.0 process model,
- map is a mapping function between the elements of the \mathcal{P} process model and the elements of the \mathbb{K}_{SKE} rule base.

The \mathbb{K}_{SKE} rule base specifies the value of *language*, such as: $\forall r_i \in R \text{ language}(r) = \text{“XTT2”}$. Moreover, the *infer*(t) method is defined as follows: $infer(t) = A_i^{dec}$. This stems from the fact that in the SKE-specific Business Logic Model, every decision component $t \in T_{\mathcal{X}}$ is an XTT2

decision table. Thus, the result of the inference is the set of decision attributes of this decision table.¹⁵

In the following paragraphs, the integration details are specified.¹⁶

Conditional Sequence Flow For the Conditional Sequence Flows $f \in \mathcal{F}_{Conditional}$ the following hold:

- All BPMN conditional sequence flows in \mathcal{P} have the condition in the form of a conditional part of a rule from the \mathbb{K}_{SKE} rule base assigned, formalized, the following holds:

$$\forall f \in \mathcal{F}_{Conditional} \exists r_i \in R (map(f) = r_i) \wedge (body(f) = E_i^{cond}) \wedge (language(f) = \text{“XTT2”}).$$

- Values of the condition attributes required by the rule are mapped to the values of corresponding attributes in the rule base:

$$\forall \begin{matrix} f \in \mathcal{F}_{Conditional} \\ r_i \in R \\ map(f) = r_i \end{matrix} \forall \lambda \in body(f) \exists a_i \in E_i^{cond} \lambda(f) \in \mathbb{D}_i \wedge \lambda(f) = a_i.$$

In the case of the data associated to the process model, we do not precise the data type (dynamic typing). However, our model implies the requirement of similar data type by checking if the value belongs to the domain of the attribute specified in rules.

Conditional Event For the Conditional Events the following hold:

- All BPMN conditional events in \mathcal{P} have the condition in the form of a conditional part of a rule from the \mathbb{K}_{SKE} rule base assigned, i.e.:

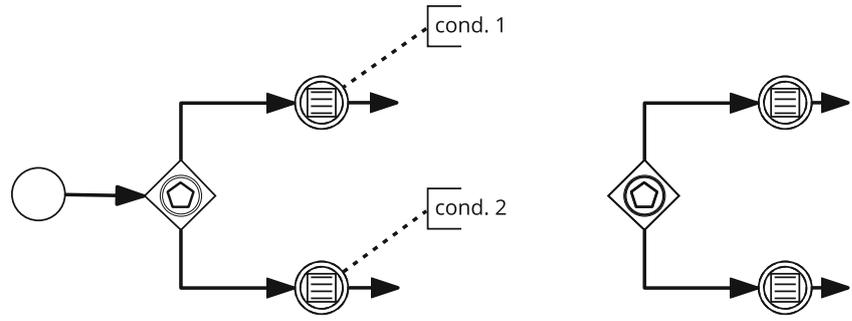
$$\forall e \in \mathcal{E}_{Conditional} \exists r_i \in R (map(e) = r_i) \wedge (body(e) = E_i^{cond}) \wedge (language(e) = \text{“XTT2”}).$$

¹⁴Note that every rule in the XTT2 representation belongs to a particular decision table. Thus, there is no rule which would not be an element of a decision table. However, it is possible that a decision table can consist of a single rule.

¹⁵More precisely: attributes and their values that are set by a particular rule. An XTT2 decision table is a first hit table (OMG 2011b), so it returns the output of a single rule (the first hit).

¹⁶If for a particular element, there are no additional requirements or conditions to specify, the formulae from General Business Logic can be used.

Fig. 10 Even-based exclusive gateways (non-instantiating and instantiating)



- Values of the condition attributes required by the rule are mapped to the values of corresponding attributes in the rule base:

$$\forall e \in \mathcal{E}_{Conditional} \quad \forall \lambda \in body(e) \exists a_i \in E_i^{cond} \quad \lambda(e) \in \mathbb{D}_i \wedge \lambda(e) = a_i. \\ r_i \in R \\ map(e) = r_i$$

Business Rule Task For the BPMN Business Rule tasks, the following formulas have to be fulfilled:

- All BPMN BR tasks in \mathcal{P} have the decision component from the \mathbb{K}_{SKE} rule base assigned:

$$\forall \tau \in \mathcal{T}_{BusinessRule} \exists t \in T_{\mathcal{X}} \quad map(\tau) = t.$$

- All the input attributes required by the HEART rule engine¹⁷ for a rule set specified by the decision component should be available in the process model, i.e.:

$$\forall \tau \in \mathcal{T}_{BusinessRule} \quad \forall a_i \in A_i^{cond} \exists \lambda \in dataInputs(\tau) \quad \lambda(\tau) \in \mathbb{D}_i \wedge \lambda(\tau) = a_i. \\ t \in T_{\mathcal{X}} \\ map(\tau) = t$$

- All the output attributes from the result of inference on a specified rule set from the HEART rule engine should be available as the output of BR task in the process, i.e.:

$$\forall \tau \in \mathcal{T}_{BusinessRule} \quad \forall \lambda \in dataOutputs(\tau) \exists a_i \in A_i^{dec} \quad \lambda(\tau) \in \mathbb{D}_i \wedge \lambda(\tau) = a_i. \\ t \in T_{\mathcal{X}} \\ map(\tau) = t$$

Diverging (Exclusive, Inclusive/Multi-choice and Complex) Gateways For the Diverging (Exclusive, Inclusive/Multi-choice and Complex) Gateways the following hold:

- All BPMN sequence flows (apart from the default ones) outgoing from a diverging gateway have the condition

in the form of a conditional part of a rule from the \mathbb{K}_{SKE} rule base assigned, i.e.:

$$\forall f \in \mathcal{F}_{g,div}^{out,cond} \exists r \in R \quad (map(f) = r_i) \wedge (body(f) = E_i^{cond}) \\ \wedge (language(f) = \text{“XTT2”}).$$

- In the case of exclusive gateways, the evaluated conditions have to be exclusive, i.e.:

$$\forall f_1, f_2 \in \mathcal{F}_{g,div}^{out} \quad \forall g \in \mathcal{G}_{\times} \quad (\exists r_1, r_2 \in R \quad map(f_1) = r_1 \wedge map(f_2) = r_2) \\ \Rightarrow (eval(r_1) \neq eval(r_2)).$$

Gateway Preceded by a BR Task For the Gateways preceded by a BR task the following hold:

- All BPMN sequence flows (apart from the default sequence flows) outgoing from a diverging gateway preceded by the BR task have the conditions based on the output attributes of the BR task, i.e.:

$$\forall \tau \in \mathcal{T}_{BusinessRule} \quad \forall g \in \mathcal{G}_{div}^{cond} \quad ((\tau, g, \lambda_{\tau g}) \in \mathcal{F}) \\ t \in T \\ map(\tau) = t \\ \Rightarrow \left(\forall f \in \mathcal{F}_{g}^{out} \forall \lambda \in body(f) \exists a_i \in A_i^{dec} \quad \lambda(\tau) \in \mathbb{D}_i \wedge \lambda(\tau) = a_i \right).$$

The whole specification of the BP Model Integrated with the XTT2 Rules with constraints defining the connections between process elements and rules was presented in the PhD thesis of the first author (Kluza 2015). This simple notation will be used in the following section for description of the case study example.

6 Case Example Described Using the Model

In order to evaluate the proposed model, we used selected use case examples which show its feasibility and efficiency. The described models are executable¹⁸ in the provided runtime environment (Nalepa et al. 2013). Full discussion of all the evaluated models was provided in Kluza (2015).

¹⁷HEART is an inference engine that is used in the SKE approach. For more information see Nalepa (2010a).

¹⁸The models consist of the BPMN 2.0 elements from the Common Executable Conformance Sub-Class (OMG 2011a).

Table 2 The names of the tasks in the example

Index	Name of the task (task labels)
BR1	Determine client class
BR2	Calculate base charge
BR3	Calculate driver discount base
BR4	Calculate car discount base
BR5	Calculate other discount base
BR6	Calculate driver discount
BR7	Calculate car discount
BR8	Calculate other discount
BR9	Calculate payment
U1	Enter car capacity information
U2	Enter Bonus Malus information
U3	Enter Premium information
U4	Display payment result

To demonstrate the application our model, and its benefits, in this section we discuss one selected case. In our opinion it is an illustrative example of a system that benefits for the integration of process and rules. On a high level, the whole decision making process can be modeled using the the BPMN process model. The lower level logic is then described with specific business rules using the XTT2 notation.

The Polish Liability Insurance (PLI) case study, was developed as one of the benchmark cases for the SKE approach for rule-based systems (Nalepa 2011). This is a case, in which the price for the liability insurance for protecting against third party insurance claims is to be calculated. The price is calculated based on various reasons, which can be obtained from the insurance domain expert. The main factors in calculating the liability insurance premium are data about the vehicle: the car engine capacity, the car age, seats, and a technical examination. Additionally, the impact on the insurance price have the driver’s age, the period of holding the license, the number of accidents in the last year, and the previous class of insurance. In the calculation, the insurance premium can be increased or

decreased because of number of payment installments, other insurances, continuity of insurance or the number of cars insured.

An excerpt of the most relevant formulas of the \mathcal{M}_{SKE}^{PLI} model is as follows (the abbreviations for the names are presented in Table 2):

$$\begin{aligned} \mathcal{M}_{SKE}^{PLI} &= (\mathcal{P}^{PLI}, \mathbb{K}_{SKE}^{PLI}, \text{map}^{PLI}), \text{ where:} \\ \mathcal{P}^{PLI} &= (\mathcal{O}, \mathcal{F}, \Lambda), \\ \mathcal{O} &= \mathcal{A} \cup \mathcal{E} \cup \mathcal{G}, \\ \mathcal{A} &= \mathcal{T}_{Business\ Rule} \cup \mathcal{T}_{User}, \\ \mathcal{T}_{Business\ Rule} &= \{\tau_{BR1}, \tau_{BR2}, \tau_{BR3}, \tau_{BR4}, \tau_{BR5}, \tau_{BR6}, \\ &\tau_{BR7}, \tau_{BR8}, \tau_{BR9}\}, \\ \mathcal{T}_{User} &= \{\tau_{U1}, \tau_{U2}, \tau_{U3}, \tau_{U4}\}, \\ \mathcal{E} &= \{e_{Start}, e_{End}\}, \\ |\mathcal{G}| &= 4. \end{aligned}$$

The process model, presented in Fig. 11, consists of: 4 User tasks, 9 Business Rule tasks, start and end events, as well as 4 parallel gateways. This model can be integrated with rules from the \mathbb{K}_{SKE}^{PLI} rule base. In such a case, the Business Rule tasks have to be associated with the decision tables from the $T_{\mathcal{X}}$ set containing the proper XTT2 rules.

Below, the specification of decision tables is presented (it provides decision table schemas which have to be complemented with XTT2 rules).

$$\begin{aligned} \mathbb{K}_{SKE}^{PLI} &= (A, R, T_{\mathcal{X}}), \text{ where:} \\ T_{\mathcal{X}} &= \{t_{BR1}, t_{BR2}, t_{BR3}, t_{BR4}, t_{BR5}, t_{BR6}, t_{BR7}, t_{BR8}, t_{BR9}\}, \\ \text{schema}(t_{BR1}) &= (\{accidentNo, clientClass\}, \\ &\{clientClass\}), \\ \text{schema}(t_{BR2}) &= (\{carCapacity\}, \{baseCharge\}), \\ \text{schema}(t_{BR3}) &= (\{clientClass\}, \{driverDiscountBase\}), \\ \text{schema}(t_{BR4}) &= (\{carAge\}, \{carDiscountBase\}), \\ \text{schema}(t_{BR5}) &= (\{installmentNo, insuranceCont, \\ &insuranceCarsNo\}, \{otherDiscountBase\}), \\ \text{schema}(t_{BR6}) &= (\{driverAge, driverLicenceAge, \\ &driverDiscountBase\}, \{driverDiscount\}), \\ \text{schema}(t_{BR7}) &= (\{seatsNo, technical, antiqueCar, \\ &carDiscountBase\}, \{carDiscount\}), \\ \text{schema}(t_{BR8}) &= (\{insuranceHistory, \\ &otherInsurance, otherDiscountBase\}, \{otherDiscount\}), \\ \text{schema}(t_{BR9}) &= (\{baseCharge, driverDiscount, \\ &carDiscount, otherDiscount\}, \{payment\}). \end{aligned}$$

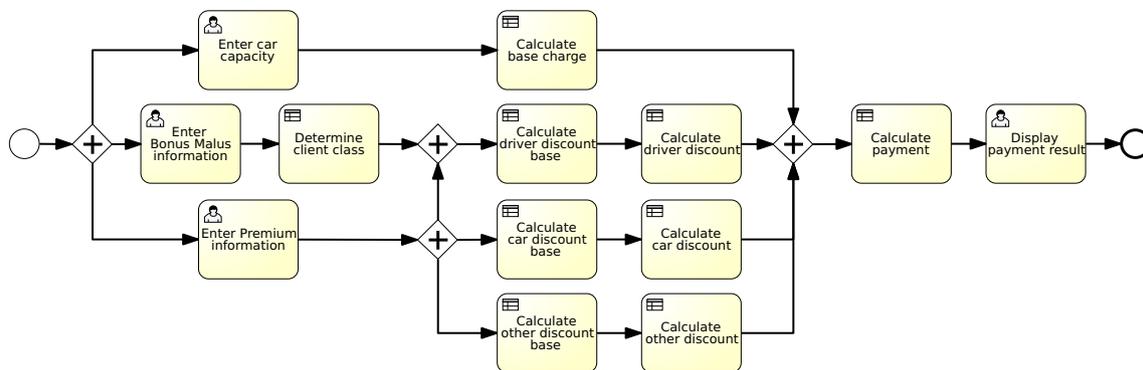


Fig. 11 The BPMN model for the PLI case study

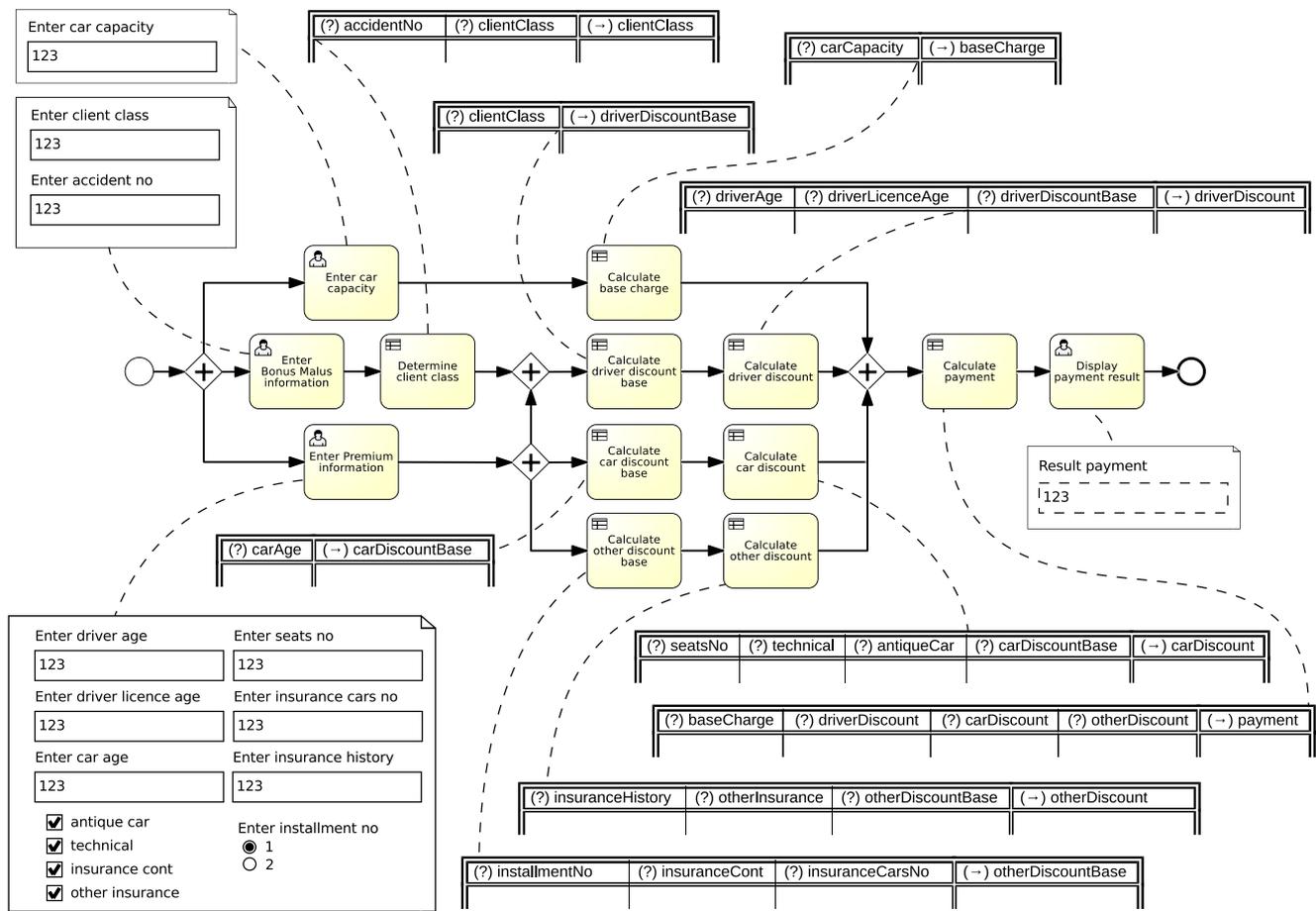


Fig. 12 The BPMN model for the PLI case study with forms and rules

$$map^{PLI} = \{(\tau_{BR1}, t_{BR1}), (\tau_{BR2}, t_{BR2}), (\tau_{BR3}, t_{BR3}), (\tau_{BR4}, t_{BR4}), (\tau_{BR5}, t_{BR5}), (\tau_{BR6}, t_{BR6}), (\tau_{BR7}, t_{BR7}), (\tau_{BR8}, t_{BR8}), (\tau_{BR9}, t_{BR9})\}.$$

The SKE-specific Business Logic Model for the PLI case study, is presented in Fig. 12. One can observe that the BR tasks in the process are connected with decision tables (for clarity in Fig. 12 only decision table schemas are presented,

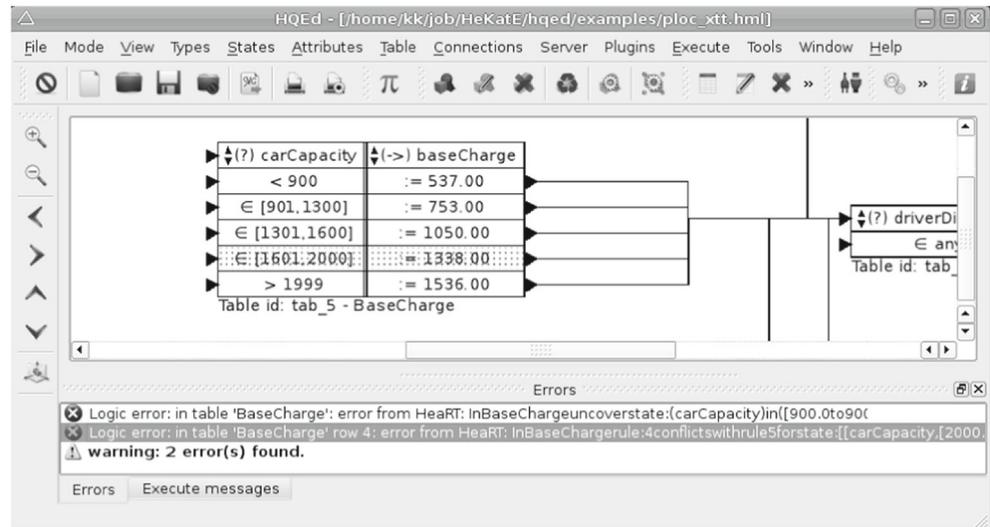
e.g. the “Calculate other discount base” Business Rule task in the process model is connected with the decision table schema: $schema(t_{BR5})$.

The decision table filled in with suitable rules is presented in Table 3). All decision tables with rules with the corresponding executable HMR representation can be found in the PhD thesis of the first author (Kluza 2015).

Table 3 “Calculate other discount base” XTT2 decision table

(?) installmentNo	(?) insuranceCont	(?) insuranceCarsNo	(- >) otherDiscountBase
1	1	1	- 10
1	0	1	0
1	1	>= 1	- 20
1	0	>= 1	- 10
2	1	1	0
2	0	1	10
2	1	>= 1	- 10
2	0	>= 1	0

Fig. 13 The design and verification process of the rule model



In the model, this decision table is represented as follows:

$$t_{BR5} = (r_1, r_2, \dots, r_8)$$

$$\begin{aligned}
 r_1 &= ((installmentNo, =, 1), (insuranceCont, =, 1), \\
 &\quad (insuranceCarsNo, =, 1)), \\
 &\quad \{(otherDiscountBase, :=, -10)\}), \\
 r_2 &= ((installmentNo, =, 1), (insuranceCont, =, 0), \\
 &\quad (insuranceCarsNo, =, 1), \\
 &\quad \{(otherDiscountBase, :=, 0)\}), \\
 &\quad \dots \\
 r_8 &= ((installmentNo, =, 2), (insuranceCont, =, 0), \\
 &\quad (insuranceCarsNo, >=, 1), \{(otherDiscountBase, :=, 0)\}).
 \end{aligned}$$

The objective of this discussion was the demonstration how the proposed approach can support the business analysts in designing process models integrated with rules. In our work we assume that there is a general formalized model of business logic. It based on the widely accepted BPMN notation, and it incorporates rule models that specify low level logic. We use the XTT2 rules, which are formalized, but also provide a visual notation for decision tables. Therefore the main benefits of our proposal include:

- a single coherent formalization of the complete model composed of processes and rules,
- support for a fully visual design, as our rules in decision tables can be designed using tools for SKE method, finally
- the XTT2 rules can be verified in the dedicated environment (Nalepa et al. 2011a) that uses the HalVA rule analysis framework for verification.

Using the XTT2 approach, the logical verification of decision tables (including completeness, determinism, redundancy, subsumption or equivalence checks) is possible

(Nalepa et al. 2011a). However, the verification issues are out of scope for this paper. The visual design and verification of the XTT2 model is shown in Fig. 13, using the HQEd editor for XTT2.

In order to show the feasibility and efficiency of the proposed approach, we have selected the non trivial use case examples that describe a business process and provide sufficient data for formulating rules. As the part of the research project,¹⁹ 9 benchmark cases were selected (see Table 4). Some of them are well-known benchmark case studies, such as: the Polish Liability Insurance case study – *PLI* (developed as a benchmark case for the SKE approach for rule-based systems (Nalepa 2011)), *EUrent* Company – *EUrent* (provided as a part of SBVR specification (OMG 2006)), and the *UServ* Financial Services case – *UServ* (a benchmark case study from Business Rules Forum²⁰). The main goal of the evaluation was to demonstrate that the proposed approach can support the design of process models integrated with rules.

The presented approach was also used in the *Prosecco* (Processes Semantics Collaboration for Companies) project²¹ finding industrial application. The project aimed at addressing the needs and constraints of small and medium enterprises (SME) by providing methods that would improve BPM systems by simplification of the system design and configuration, targeting the management quality and competitiveness improvement. In the project a set of use cases was acquired from the involved SMEs. These cases were composed of the dominating process part, but were augmented with rules specifying low level decision logic. We use our integrated model to design these

¹⁹See: <http://geist.agh.edu.pl/pub/projects:hibuproburul:start>.

²⁰See: <http://www.businessrulesforum.com>.

²¹See: <http://prosecco.agh.edu.pl>.

Table 4 Comparison of the selected cases

Case name	$ A $	$ \mathcal{T}_{User} $	$ \mathcal{T}_{BR} $	$ \mathcal{G} $	$ R $
Upsell	5	2	3	0	15
Hello	7	3	3	2	8
Health	8	2	3	2	8
Thermostat	7	2	4	4	18
CashPoint	12	3	5	4	11
Cardio	15	2	7	4	18
EURent	26	6	7	4	31
sPLI	22	4	9	4	61
UServ	42	6	19	17	84

heterogeneous cases. Furthermore, the runtime environment we provided was able to run these integrated models. In this case the BP runtime engine (Activity) delegated the execution of the rule components to our dedicated embedded rule engine (HearTDroid).

7 Conclusions

The main contribution of this paper is a new model for integration of Business Processes with Business Rules. This model is based on existing representation methods for processes (the BPMN notation) and rules (the XTT2 representation). The model is fully formalized. It uses and extends existing formalizations of BPMN, as well as our previous formalization of XTT2 rules. Furthermore, the model supports the business analysts in the design of process models integrated with rules.

In the paper we provided motivation for our work, as well as positioned it the area of related works. We presented demonstration of the application of the model on a selected use case. The evaluation demonstrated that the presented model provides adequate formal means for describing a process model integrated with rules. In our opinion important benefits of our proposal include: a uniform formalization of processes and rules, support for a visual design of integrated model, as well as opportunities for their formal verification. Moreover, from rule-based systems point of view, such a model can be treated as a structured rule base that provides explicit inference flow determined by the process control flow.

The presented model can be used for a clear logical description of a process model, especially for specification of integration issues and ensuring data types consistency. It was used in specification of the algorithm for generation of the integrated models from the ARD diagrams (Kluza et al. 2015). In fact, it constitutes the base for our method for generation and design of Business Processes with Business Rules presented in Kluza and Nalepa (2017). As in the

provided model, the method uses the integrated model of the existing representations for processes and rules, such as the BPMN notation for process models, and the XTT2 representation for rules.

Moreover, the presented XTT2 rules can be formally verified using the dedicated rule analysis framework (Nalepa et al. 2011a). In the case of the process model, it is possible to use some of the existing verification methods for processes.

The model can also be used as a specification of constraints for execution purposes. As the BPMN models are executable in process engines and rules in the XTT2 representation can be executed in the HEART rule engine, such integrated models can be executed in the hybrid runtime environment (Nalepa et al. 2013).

As in the DMN specification linking BPMN and DMN models is described informally, our future works will be focused on adjusting our approach to support the DMN notation. One of the directions is to extend our model to support DMN decision tables and provide rules interoperability between these methods (Kaczor 2015). Thus, it will be possible to take advantage of the formal verification issues available in our approach. Another research direction concerns the advantages of the transforming our method of generation models (Kluza and Nalepa 2017) which is based on the model described in this paper, in order to support DMN. Especially, this will focus on transforming our representation to DMN (similarly to the transformation of PDM (van der Aa et al. 2016)).

Finally, our rule-based representation has several important extension which (to the best of our knowledge) are not available for DMN. The most important regards uncertainty handling on the rule level. In our future work we plan to explore the possible introduction of uncertainty handling to the integrated decision model.

Acknowledgements The research presented in this paper is partially based on the PhD thesis (Kluza 2015) of the first author and is supported by the HiBuProBuRul Project funded from NCN (National Science Centre) resources for science (no. DEC-2011/03/N/ST6/00909).

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

- Bădică, A., & Bădică, C. (2011). Formal verification of business processes as role activity diagrams. In *2011 Federated Conference on computer science and information systems (FedCSIS)* (pp. 277–280): IEEE.

- Bădică, C., Bădică, A., Litoiu, V. (2003). Role activity diagrams as finite state processes. In *Proceedings of the second international conference on parallel and distributed computing ISPD'03, Ljubljana, Slovenia, October 13–14, 2003* (pp. 15–22).
- Batoulis, K., Baumgrass, A., Herzberg, N., Weske, M. (2015a). Enabling dynamic decision making in business processes with DMN. In *International conference on business process management* (pp. 418–431): Springer.
- Batoulis, K., Meyer, A., Bazhenova, E., Decker, G., Weske, M. (2015b). Extracting decision logic from process models. In *International conference on advanced information systems engineering* (pp. 349–366): Springer.
- Bazhenova, E., & Weske, M. (2015). Deriving decision models from process models by enhanced decision mining. In *International conference on business process management* (pp. 444–457): Springer.
- Catalkaya, S., Knuplesch, D., Chiao, C., Reichert, M. (2013). Enriching business process models with decision rules. In *International conference on business process management* (pp. 198–211): Springer.
- Changizi, B., Kokash, N., Arbab, F. (2010). A unified toolset for business process model formalization. In *Proceeding of the 7th international workshop on formal engineering approaches to software components and architectures, satellite event of ETAPS, held on 27th March 2010, Paphos, Cyprus* (p. 10).
- Charfi, A., & Mezini, M. (2004). Hybrid web service composition: business processes meet business rules. In *Proceedings of the 2nd international conference on service-oriented computing, ICSOC '04* (pp. 30–38). New York: ACM.
- Debevoise, T., Taylor, J., Sinur, J., Geneva, R. (2014). The MicroGuide to process and decision modeling in BPMN/DMN: building more effective processes by integrating process modeling with decision modeling. CreateSpace Independent Publishing Platform.
- Dijkman, R.M., & Gorp, P.V. (2011). Bpmn 2.0 execution semantics formalized as graph rewrite rules. In Mendling, J., Weidlich, M., Weske, M. (Eds.) *Proceedings from the business process modeling notation—second international workshop, BPMN 2010, Potsdam, Germany, October 13–14, 2010. Lecture notes in business information processing*, (Vol. 67 pp. 16–30): Springer.
- Dijkman, R.M., Dumas, M., Ouyang, C. (2007). Formal semantics and automated analysis of BPMN process models. preprint 7115. Tech. rep. Queensland University of Technology, Brisbane, Australia.
- Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A. (2013). *Fundamentals of business process management*. Berlin: Springer.
- Hohwiller, J., Schlegel, D., Grieser, G., Hoekstra, Y. (2011). Integration of BPM and BRM. In Dijkman, R., Hofstetter, J., Koehler, J. (Eds.) *Business process model and notation, Lecture notes in business information processing*, (Vol. 95 pp. 136–141). Berlin: Springer. https://doi.org/10.1007/978-3-642-25160-3_12.
- Janssens, L., Bazhenova, E., De Smedt, J., Vanthienen, J., Denecker, M. (2016). Consistent integration of decision (DMN) and process (BPMN) models. In *Proceedings of the CAiSE'16 forum, at the 28th international conference on advanced information systems engineering (CAiSE 2016), Ljubljana, Slovenia, June 13–17, 2016* (pp. 121–128).
- Kaczor, K. (2015). Practical approach to interoperability in production rule bases with SUBITO. In Rutkowski, L. et al. (Eds.) *Artificial intelligence and soft computing: 14th international conference, ICAISC 2015, Lecture notes in artificial intelligence*. Zakopane: Springer. Accepted.
- Kluza, K. (2015). Methods for modeling and integration of business processes with rules. PhD thesis, AGH University of Science and Technology, Supervisor: Grzegorz J. Nalepa.
- Kluza, K., & Nalepa, G.J. (2017). A method for generation and design of business processes with business rules. *Information and Software Technology*, 91, 123–141.
- Kluza, K., Kaczor, K., Nalepa, G.J. (2012). Enriching business processes with rules using the Oryx BPMN editor. In Rutkowski, L. et al. (Eds.) *Artificial intelligence and soft computing: 11th international conference, ICAISC 2012: Zakopane, Poland, April 29–May 3, 2012, Lecture notes in artificial intelligence*, (Vol. 7268 pp. 573–581): Springer. <http://www.springerlink.com/content/u654r0m56882np77/>.
- Kluza, K., Kaczor, K., Nalepa, G.J. (2015). Integration of business processes with visual decision modeling. Presentation of the hades toolchain. In Fournier, F., & Mendling, J. (Eds.) *Business process management workshops. BPM 2014 international workshops, Eindhoven, The Netherlands, September 7–8, 2014, Revised Papers, Lecture notes in business information processing*, (Vol. 202 pp. 504–515): Springer International Publishing. https://doi.org/10.1007/978-3-319-15895-2_43.
- Knolmayer, G., Endl, R., Pfahrer, M. (2000). Modeling processes and workflows by business rules. In *Business process management, models, techniques, and empirical studies* (pp. 16–29). London: Springer.
- Lam, V.S.W. (2009). Equivalences of BPMN processes. *Service Oriented Computing and Applications*, 3(3), 189–204.
- Lam, V.S.W. (2012). Foundation for equivalences of BPMN models. *Theoretical and Applied Informatics*, 24(1), 33–66.
- Ligeża, A. (1999). Intelligent data and knowledge analysis and verification; towards a taxonomy of specific problems. In Vermesan, A., & Coenen, F. (Eds.) *Validation and verification of knowledge based systems* (pp. 313–325): Springer. https://doi.org/10.1007/978-1-4757-6916-6_21.
- Ligeża, A. (2006). *Logical foundations for rule-based systems*. Berlin: Springer.
- Ligeża, A. (2011). BPMN – a logical model and property analysis. *Decision Making in Manufacturing and Services*, 5(1-2), 57–67.
- Lindsay, A., Dawns, D., Lunn, K. (2003). Business processes – attempts to find a definition. *Information and Software Technology*, 45(15), 1015–1019.
- Nalepa, G.J. (2010a). Architecture of the HeaRT hybrid rule engine. In Rutkowski, L. et al. (Eds.) *Artificial intelligence and soft computing: 10th international conference, ICAISC 2010: Zakopane, Poland, June 13–17, 2010, Pt. II, Lecture notes in artificial intelligence*, (Vol. 6114 pp. 598–605): Springer.
- Nalepa, G.J. (2010b). HeKatE methodology, hybrid engineering of intelligent systems. *International Journal of Applied Mathematics and Computer Science*, 20(1), 35–53.
- Nalepa, G.J. (2011). *Semantic knowledge engineering a rule-based approach*. Kraków: Wydawnictwa AGH.
- Nalepa, G.J., & Ligeża, A. (2005). Software engineering: evolution and emerging technologies, frontiers in artificial intelligence and applications. In *Conceptual modelling and automated implementation of rule-based systems*, (Vol. 130 pp. 330–340). Amsterdam: IOS Press.
- Nalepa, G., Bobek, S., Ligeża, A., Kaczor, K. (2011a). HalVA – rule analysis framework for XTT2 rules. In Bassiliades, N., Governatori, G., Paschke, A. (Eds.) *Rule-based reasoning, programming, and applications, Lecture notes in computer science*, (Vol. 6826 pp. 337–344). Berlin: Springer. <http://www.springerlink.com/content/c276374nh9682jm6/>.
- Nalepa, G., Ligeża, A., Kaczor, K. (2011b). Overview of knowledge formalization with XTT2 rules. In Bassiliades, N., Governatori, G., Paschke, A. (Eds.) *Rule-based reasoning, and programming, and applications, Lecture notes in computer science*, (Vol. 6826 pp. 329–336). Berlin: Springer.

- Nalepa, G.J., Ligęza, A., Kaczor, K. (2011c). Formalization and modeling of rules using the XTT2 method. *International Journal on Artificial Intelligence Tools*, 20(6), 1107–1125.
- Nalepa, G.J., Kluza, K., Kaczor, K. (2013). Proposal of an inference engine architecture for business rules and processes. In Rutkowski, L. et al. (Eds.) *Artificial intelligence and soft computing: 12th international conference, ICAISC 2013: Zakopane, Poland, June 9–13, 2013, Lecture notes in artificial intelligence*, (Vol. 7895 pp. 453–464): Springer. <http://www.springer.com/computer/ai/book/978-3-642-38609-1>.
- OMG (2006). Semantics of business vocabulary and business rules (SBVR). Tech. Rep. dtc/06-03-02, Object Management Group.
- OMG (2011a). Business process model and notation (BPMN): Version 2.0 specification. Tech. Rep. formal/2011-01-03, Object Management Group.
- OMG (2011b). Decision model and notation request for proposal. Tech. Rep. bmi/2011-03-04, Object Management Group, 140 Kendrick Street, Building A Suite 300, Needham, MA 02494, USA.
- OMG (2015). Decision model and notation (DMN). Version 1.0: Formal Specification. Tech. rep., Object Management Group (OMG). <http://www.omg.org/spec/DMN/1.0/Beta2/>.
- Ouyang, C., Wil, M.P., van der Aalst, M.D., ter Hofstede, A.H. (2006a). Translating BPMN to BPEL. Tech. rep., Faculty of Information Technology, Queensland University of Technology, GPO Box 2434, Brisbane QLD 4001, Australia Department of Technology Management, Eindhoven University of Technology, GPO Box 513, NL-5600 MB, The Netherlands.
- Ouyang, C., Dumas, M., ter Hofstede, A.H., van der Aalst, W.M. (2006b). From BPMN process models to BPEL web services. In *IEEE international conference on web service (ICWS'06)*.
- Sivaraman, E., & Kamath, M. (2002). On the use of Petri nets for business process modeling. In *Proceeding of the 11th annual industrial engineering research conference*.
- Smirnov, S., Weidlich, M., Mendling, J., Weske, M. (2012). Action patterns in business process model repositories. *Computers in Industry*, 63(2), 98–111.
- Speck, A., Feja, S., Witt, S., Pulvermüller, E., Schulz, M. (2011). Formalizing business process specifications. *Computer Science and Information Systems/ComSIS*, 8(2), 427–446.
- Szpyrka, M., Nalepa, G.J., Ligęza, A., Kluza, K. (2011). Proposal of formal verification of selected BPMN models with Alvis modeling language. In Brazier, F.M., Nieuwenhuis, K., Pavlin, G., Warnier, M., Badica, C. (Eds.) *Intelligent distributed computing V. Proceedings of the 5th international symposium on intelligent distributed computing – IDC 2011, Delft, the Netherlands – October 2011, Studies in computational intelligence*, (Vol. 382 pp. 249–255): Springer. <http://www.springerlink.com/content/m181144037q67271/>.
- van der Aa, H., Leopold, H., Batoulis, K., Weske, M., Reijers, H.A. (2016). Integrated process and decision modeling for data-driven processes. In *Business process management workshops. BPM 2015 international workshops, Innsbruck, Austria, August 31–September 3, 2015, Revised Papers, Lecture notes in business information processing* (pp. 405–417): Springer International Publishing. https://doi.org/10.1007/978-3-319-42887-1_33.
- Wagner, G., Giurca, A., Lukichev, S. (2005). R2ML: a general approach for marking up rules. In Bry, F., Fages, F., Marchiori, M., Ohlbach, H. (Eds.) *Principles and practices of semantic web reasoning, Dagstuhl seminar proceedings* (p. 05371).
- Wang, W., Indulska, M., Sadiq, S., Weber, B. (2017). Effect of linked rules on business process model understanding. In *International conference on business process management*, (Vol. 10445, pp. 200–215). Springer.
- Weske, M. (2012). *Business process management: concepts, languages, architectures*, 2nd edn. Berlin: Springer.
- Wong, P.Y.H., & Gibbons, J. (2008). A process semantics for BPMN. In Liu, S., Maibaum, T.S.E., Araki, K. (Eds.) *Proceedings from the 10th international conference on formal engineering methods, ICFEM 2008, Kitakyushu-City, Japan, October 27–31, 2008, Lecture notes in computer science*, (Vol. 5256 pp. 355–374): Springer.
- Wong, P.Y.H., & Gibbons, J. (2011). Formalisations and applications of BPMN. *Science of Computer Programming*, 76(8), 633–650.
- Wynn, M., Verbeek, H., Wvd, A., At, H., Edmond, D. (2009). Business process verification – finally a reality!. *Business Process Management Journal*, 1(15), 74–92.

Krzysztof Kluza is an assistant professor at the AGH University of Science and Technology in Krakow. Currently, he is Deputy Dean of Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering at AGH UST. He obtained MSc in Automatics & Robotics (2009) at AGH UST, MA in Cultural Studies (2010) at Jagiellonian University in Krakow, and PhD in Computer Science (2015) at AGH UST. Krzysztof Kluza is also an alumnus of “Top 500 Innovators” Science-Management-Commercialization program at Stanford University and a Vice President of the Creativity and Innovation Lab Foundation. He published over 50 papers related to knowledge and software engineering. His primary scientific interests focus on business processes and business rules in information systems.

Grzegorz J. Nalepa is an engineer with degrees in computer science - artificial intelligence, and philosophy. He has been working in the area of intelligent systems and knowledge engineering for over 15 years. He formulated the eXtended Tabular Trees rule representation method, as well as the Semantic Knowledge Engineering approach. He authored a book “Modeling with Rules using Semantic Knowledge Engineering” (Springer 2018). He co-edited a book “Synergies Between Knowledge Engineering and Software Engineering” (Springer 2018). He co-authored over 150 research papers in international journals and conferences. He coordinates GEIST - Group for Engineering of Intelligent Systems and Technologies (<http://geist.re>), and cooperates with AGH University and Jagiellonian University in Krakow, Poland. He is a member of Polish Artificial Intelligence Society (PSSI), IEEE, Italian Artificial Intelligence Society (AI*IA) and KES. His recent interests include context-aware systems and affective computing.