Validation of User Intentions in Process Models

Gerd Gröner[1], Mohsen Asadi[2], Bardia Mohabbati[2], Dragan Gašević[3], Fernando Silva Parreiras[4], and Marko Bošković[3]

[1] WeST Institute, University of Koblenz-Landau, Germany
groener@uni-koblenz.de
[2] Simon Fraser University, Canada
{masadi,mohabbati}@sfu.ca
[3] Athabasca University, Canada
{dragang,marko.boskovic}@athabascau.ca
[4] FUMEC University, Brazil
fernando.parreiras@fumec.br

Abstract. Goal models and business process models are complementary artifacts for capturing the requirements and their execution flow in software engineering. Usually, goal models serve as input for designing business process models, and it requires mappings between both types of models. Due to the large number of possible configurations of elements from both goal models and business process models, developers struggle with the challenge of maintaining consistent configurations of both models and their mappings. Managing these mappings manually is error-prone. In our work, we propose an automated solution that relies on Description Logics and automated reasoners for validating mappings that describe the realization of goals by activities in business process models. The results are the identification of two inconsistency patterns – strong inconsistency and potential inconsistency, and the development of the corresponding algorithms for detecting inconsistencies.

Keywords: requirement modeling, goal-oriented process engineering, inconsistency detection.

1 Introduction

With the growing importance of process-aware information systems (PAISs), business process modeling has gained a significant research attention [1]. A business process model is an operational representation of activities, their ordering and routing in achieving goals; that is, delivering services to customers. However, business process modeling is not an effective way to understand and elicit user requirements and intentions in the development life-cycle of PAISs.

Requirements engineering offers proven means for understanding user intentions. Goal-oriented modeling is a prominent formalism to describe requirements of a system in terms of goals and relationships (constraints) between goals. A goal describes a certain system functionality or property that should be achieved. It is

J. Ralyté et al. (Eds.): CAiSE 2012, LNCS 7328, pp. 366–381, 2012.

not surprising then that the recent research on PAISs has focused on integration of business process modeling with goal-oriented modeling (c.f. Sect. 7). Related research concentrates on basic mapping and alignment principles between goal models [2,3,4] or business models [5,6] and process models. Their main focus is either on enriching a process model with goals and relationships between goals or on transformations between goal models and process models. However, less attention has been paid to validation formalisms for the correctness of the mappings between goal-oriented models and business process models.

In this paper, we tackle the challenge of automated validation of the correctness of mappings between goal models and business process models. Such mappings define which parts of a business process model are responsible for the realization of a certain user goal. In that sense, an automated validation of the mappings needs to assure satisfaction of user goals in each execution configuration/path of business processes.

To address the challenge of the mapping validation, our first contribution (Sect. 3) is the definition of the types of inconsistencies between goal and process models based on the correspondence between intentional relationships and workflow patterns [7]. Next, based on these correspondences, we propose a modeling (Sect. 4) and validation (Sect. 5) approach in Description Logics. Our approach ensures realization equivalence between mapped goals and their corresponding activities; that is, there are no contradictions between relations of goals compared to their mapped activities.

2 Foundations

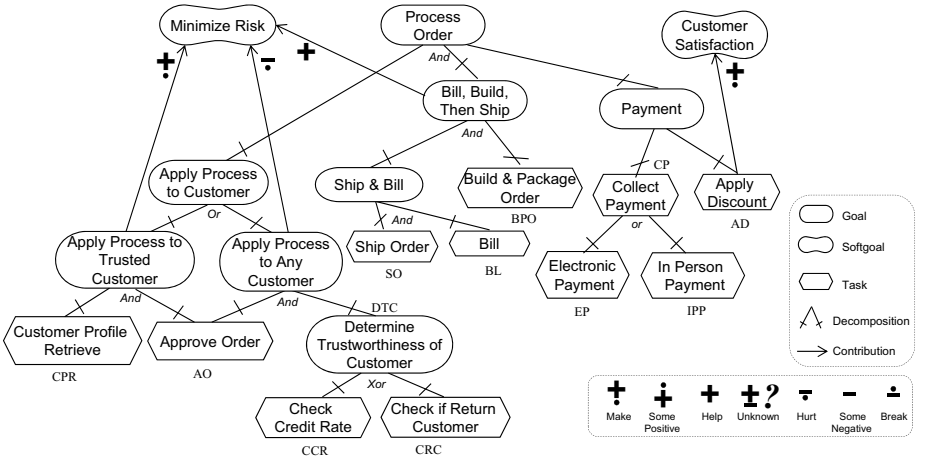
This section starts with background information on the requirement perspective and the design perspective of business process models. Furthermore, the notion of goal realization and the consequential problem statement is presented.

2.1 Goal Models

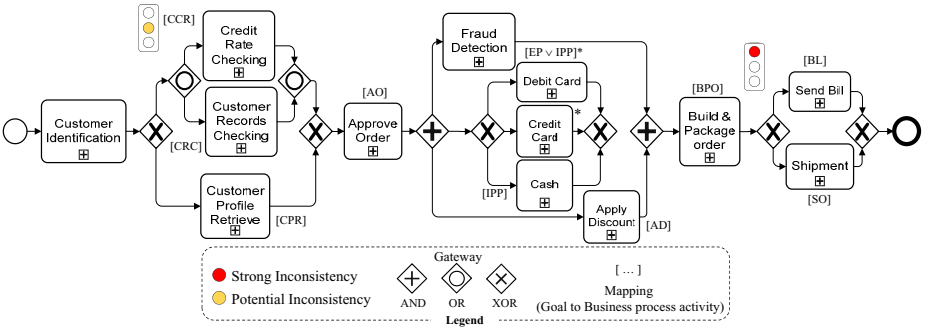
To date, several languages for goal models have been proposed, e.g., i*/Tropos [8,9], NFR [10], KAOS [11] and Goal Requirements Language (GRL) [12]. We use the GRL, a part of the recent Recommendation of the International Telecommunications Union named User Requirements Notation (URN) [12]. GRL is a language that integrates core concepts of i* and NFR [13].

A goal model is a graph, consisting of *actors*, *intentional elements*, *links* and *decompositions* [12]. *Actors* are entities that can have intentions and carry out actions. Typically, they are stakeholders or systems.[1] *Intentional elements* used in this paper are *(hard) goals*, *soft goals* and *tasks*. *Soft goals* are similar to hard

[1] In this paper, however, we are interested in validation of user intentions in service orchestrations, typically achieved within one actor. Therefore, in the later text we do not consider satisfaction levels of actors nor dependencies.



(a) Goal Model of the E-Store



(b) Business Process Model of the E-Store

Fig. 1. Goal Realizations in Business Process Models

goals, but without a clear-cut criteria for whether the condition is achieved. They model non-functional requirements[2]. *Tasks* specify conceptual solutions.

A concrete goal model is depicted in Fig. 1(a). Links connect intentional elements. They can be either *decompositions* or *contributions*. *Decompositions* allow for a specification of what source intentional elements need to be satisfied in order to satisfy the target intentional elements. For instance, the goal Ship & Bill is achieved if both tasks Ship Order and Bill are fulfilled. GRL supports *AND*, *IOR* and *XOR* decompositions. An *AND* decomposition specifies that all source intentional elements need to be satisfied for the target intentional element to be satisfied. *IOR* is used to specify that the satisfaction of at least

[2] Some non-functional requirements, like security, can have clear cut criteria and can also be achieved with different operationalizations. The GRL standard does not specify how to model these situations. Nevertheless, the standard allows for specification of decompositions on soft goals.

one source satisfies the target, whereby *XOR* specifies that exactly one of the source elements is necessary to satisfy the target.

Contribution types can be seen in Fig. 1(a). The Make and Break contributions are respectively positive and negative, and sufficient for satisfaction of a target element. Help and Hurt are also respectively positive and negative, but insufficient. The extent of the contribution of SomePositive and SomeNegative is unknown. Finally, for the Unknown contribution link, both the extent and degree (positive or negative) of the contribution are unknown.

Definition 1 (Goal Model). *A goal model is a triple* $GM = (\mathcal{G}, \mathcal{C}, \mathcal{D})$. $\mathcal{G}$ *is a set of goals (also called intentions or intentional elements). Intentions are (hard) goals* ($\mathcal{G}_g$)*, tasks* ($\mathcal{G}_t$) *and soft goals* ($\mathcal{G}_s$)*.* $\mathcal{C}$ *denotes positive and negative contributions* ($\mathcal{G} \times \{\overset{+}{\bullet}, \overset{\bullet}{+}, +, \pm?, \overset{-}{\bullet}, -, \overset{\bullet}{-}\} \times \mathcal{G}$)*.* $\mathcal{D}$ *is a decomposition of intentional elements* $\mathcal{G} \times \{IOR, XOR, AND\} \times \mathcal{P}(\mathcal{G})$*, whereby an intention* $G \in \mathcal{G}$ *is fulfilled either if at least one, exactly one or all source intentions are fulfilled.*

2.2 Business Process Models and Workflow Patterns

There is an emergence and proliferation of process-oriented software development methods for enterprises, where software is designed, built, executed by process engines, maintained and evolved on the basis of business process models [1,14]. A business process can be considered as a set of ordered activities intended to realize and implement a goal [15] according to requirement models.

Business process models can be designed at different levels of abstraction. Currently, a number of graph-based business process modeling languages exists, e.g., BPMN, EPC, YAWL and UML Activity Diagram. Although the proposed solution in this paper is generic, we use here the Business Process Modeling Notation (BPMN). Despite of variance in expressiveness and modeling notations, all modeling languages share the common concepts of activities, gateways or routing nodes, artifacts and relations between them represented as control flow.

A business process model is illustrated in Fig. 1(b). Mappings between activities and tasks in the goal model are indicated by activity annotations. E.g., the activity Credit Rate Checking is mapped to goal Check Credit Rate (CCR).

Definition 2 (Business Process Model). *A business process model is defined as a connected graph* $G_{PM}=(\mathcal{V}, \mathcal{E})$*, where* $\mathcal{V}$ *depicts a set of vertices including a set of activities* $\mathcal{A}$ *and gateways* $\mathcal{G}$*. A gateway has a type* $T(G)$ *such that* $T(G) \in \{xor, or, and, disc\}$*.* $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ *denotes a set of edges between vertices.*

Definition 3 (Materialized Business Process Model). *Given a business process model* $G_{PM} = (\mathcal{V}, \mathcal{E})$*, a materialized process model PM is a quadruple* $(\mathcal{V}, \mathcal{E}, \mathcal{F}, \mathcal{E}_A)$*. The set* $\mathcal{F} \subseteq V \times V \times \mathcal{B}$ *represents single-entry-single-exit (SESE) fragments, in which* $\mathcal{B}$ *represents a set of branches between entry and exit vertex. A branch* $B \in \mathcal{B}$ *is considered as a set of activities.* $\mathcal{E}_A$ *is a set of sequence edges that are obtained from the process graph by treating gateways as transparent.*

SESE-fragments can be derived from a process model in linear time (cf. [16]). We consider structural well-formedness conditions for business process models [17,18,19]. Gateways have either exactly one incoming edge and multiple outgoing edges or multiple incoming edges and exactly one outgoing edge. The set $\mathcal{E}_A$ is derived from $\mathcal{E}$ by neglecting gateways and replacing them by their corresponding next predecessor or successor activity. Workflow patterns [20] describe structures and behavior of processes for the execution. They are defined in terms of how the process flow proceeds in sequences and splits into branches for executing the activities and how they converge. We select the main patterns of the Workflow Patterns framework[3] as a reference analysis framework.

3 Realization Inconsistencies

In goal-oriented requirements engineering, tasks are considered requirements if they are assigned into a system-to-be. To realize requirements (i.e., tasks in the goal model), activities (either atomic or composite, i.e., sub-processes) are defined in business process models. The execution relations between activities (i.e., workflow patterns) in business process models must be consistent with intentional relations between tasks and other intentional elements in goal models.

Definition 4 (Realization Equivalence). *Assume activities $A_1, \ldots, A_n \in \mathcal{A}$ are realizations of intentional elements $G_1, \ldots, G_m \in \mathcal{G}_t$. A workflow pattern WF exists between activities $A_1, \ldots, A_n$, and an intentional relation IR exists between intentional source elements $G_1, \ldots, G_m$ and an intentional target element $G \in \mathcal{G}$. WF is realization equivalent to IR, if all execution combinations of activities in WF lead to the satisfaction of target goal G.*

If WF is defined as a realization of IR in a business process model and there is no realization equivalence between these two relations, an inconsistency in the process model can occur with respect to the goal model. We define two types of inconsistencies: 1) *strong inconsistency* and 2) *potential inconsistency.*

Definition 5 (Strong Inconsistency). *Assume a workflow pattern WF is specified over activities $A_1, \ldots, A_n \in \mathcal{A}$ and an intentional relation IR with target element $G \in \mathcal{G}$ is defined over intentional elements $G_1, \ldots, G_m \in \mathcal{G}_t$ (e.g., an AND decomposition G is a parent intentional element and the rest are children). Activities $A_1, \ldots, A_n$ are realizations of intentional elements $G_1, \ldots, G_m$. A strong inconsistency between WF and IR occurs if there is no execution combination of activities that leads to the fulfillment of the target element G.*

Definition 6 (Potential Inconsistency). *Assume a workflow pattern WF is specified over activities $A_1, \ldots, A_n \in \mathcal{A}$ and an intentional relation IR with target element $G \in \mathcal{G}$ is defined over intentional elements $G_1, \ldots, G_m \in \mathcal{G}_t$. Activities $A_1, \ldots, A_n$ are realizations of intentional elements $G_1, \ldots, G_m$. We define*

[3] `http://www.workflowpatterns.com`

a potential inconsistency between WF and IR if some execution combinations of activities lead to the fulfillment of intentional element G and some execution combinations of activities do not lead to the fulfillment of G.

The example in Fig. 1 contains a strong and a potential inconsistency. The strong inconsistency is due to the activities Send Bill and Shipment that are mapped to tasks Bill (BL) and Ship Order (SO), whereby these tasks are AND-siblings. Thus, each satisfaction of the target element Ship & Bill requires that both tasks Bill (BL) and Ship Order (SO) are fulfilled simultaneously, while each process execution allows either the execution of Send Bill or Shipment.

Table 1. Correspondence between Intentional Relations and Workflow Patterns

Workflow Patterns	Intentional Relations				
	AND	IOR	XOR	$\dot{+}$	$\dot{\bullet}$
AND-AND Parallel split - Synchronization	✓	✓	↯	✓	↯
AND-OR Parallel split - Multi merge	✓	✓	↯	✓	↯
AND-DISC Parallel split - Discriminator	✓	✓	↯	✓	↯
AND-XOR Parallel split - Simple merge	✓	✓	↯	✓	↯
OR-OR Multi choice - Synchronizing merge	±	✓	±	±	±
OR-DISC Multi choice - Discriminator	±	✓	±	±	±
OR-XOR Multi choice - Simple merge	±	✓	±	±	±
XOR-XOR Exclusive - Simple merge	↯	✓	✓	±	✓
Sequence	✓	✓	↯	✓	↯

A potential inconsistency is caused by the mapping of activities Credit Rate Checking and Customer Records Checking to the exclusive sibling tasks Check Credit Rate (CCR) and Check if Return Customer (CRC). There are executions where both activities are executed, but this would contradict to the exclusiveness of the tasks Check Credit Rate (CCR) and Check if Return Customer (CRC) to fulfill their target goal DTC.

Combinations of intentional relations (IR) and workflow patterns (WF) are shown in Table 1. If a workflow pattern WF is realization equivalent to an intentional relation IR, the corresponding cell is marked with ✓. Strong and potential inconsistencies are shown with (↯) and (±), respectively. If a set of workflow patterns $WF_1, WF_2, \ldots, WF_n$ are realizations of an intentional relation IR, then all combinations $(WF_1, IR), \ldots, (WF_n, IR)$ should be realization equivalent.

Regarding the mappings between the goal models and business process models, we consider two assumptions: i) Only tasks in the goal model are mapped to atomic activities or composite activities (sub-processes) in the business process models, since only tasks are operationalizations and realizations; thus, hard goals and soft goals are not mapped to activities. ii) If there are unmapped activities within a business process model, then those activities do not contribute to any realization.

4 Knowledge Base for Realization Validation

We use Description Logics (DL) [21] to model workflow patterns, intentional relations and mappings. Based on this representation, DL reasoning services are used to validate realizations of intentional elements by workflow patterns.

4.1 Foundations of Description Logics

DL is a decidable subset of first-order logic (FOL). A DL knowledge base consists of a TBox (Terminological Box) and an ABox (Assertional Box). The TBox is used to specify concepts, which denote sets of individuals and roles defining binary relations between individuals. The main syntactic constructs are depicted in Table 2, supplemented by the corresponding FOL expressions. Concept inclusion axioms $C \sqsubseteq D$ mean that each individual of the concept C is also an individual of D. There are two special concepts in Description Logics, namely the universal concept (top concept) $\top$ and the bottom concept $\bot$. The top concept $\top$ is the superconcept of all concepts, i.e., $C \sqsubseteq \top$ holds for each concept C. $\bot$ is an unsatisfiable concept. A concept equivalence (or definition) $C \equiv D$ is an abbreviation for two concept inclusion axions $C \sqsubseteq D$ and $D \sqsubseteq C$.

Inference services of DL rely on the well defined Tarski-style semantics. The subset of DL constructs we use in our models ($\mathcal{ALC}$ expressiveness) in combination with existing highly optimized reasoning algorithms and systems allow for practical efficient reasoning support. In the remainder of this paper, we use subsumption checking and concept classification as basic reasoning services.

4.2 Representation of Models and Realizations

The key part of our modeling formalism contains the relations of both models, combined with mappings between them. The goal model describes *intentional relations* between goals and the business process model specify *control flow relations* on activities, which refer to basic *workflow patterns* [7,20].

Representation of Intentional Relations. The *intentional relations* of a goal model GM are described in a DL knowledge base Σ_{GM}. Algorithm 1 depicts the representation of intentional relations of a goal model $GM = (\mathcal{G}, \mathcal{C}, \mathcal{D})$ in a DL knowledge base Σ_{GM}. Only tasks are realized by activities. Thus, the algorithm introduces for each task G ($G \in \mathcal{G}_t$) a concept Rel_G to capture its intentional relations, which are either decompositions or contributions. Intentional elements G are also concepts in DL.

Lines 2–4 treat IOR-decompositions of an intention into subgoals G_i. Thus, intentions G_i are disjunctively related to each other. This is represented in DL by a concept union over all intentions G_j that are members of the IOR-decomposition. For each intention G_i that is part of the IOR-decomposition, we introduce a concept Rel_{G_i} to describe the relations of each intention. In this vein, a conjunctive decomposition (lines 5–7) is described by a concept intersection in DL. As in the

Table 2. Constructs and Notations in DL and FOL Syntax

Construct Name	**DL Syntax**	**FOL Syntax**
atomic concept, atomic role	C, R	$C(x)$, $R(x,y)$
concept inclusion axiom	$C \sqsubseteq D$	$\forall x.C(x) \rightarrow D(x)$
concept union	$C_1 \sqcup \ldots \sqcup C_n$	$C_1(x) \vee \ldots \vee C_n(x)$
concept intersection	$C_1 \sqcap \ldots \sqcap C_n$	$C_1(x) \wedge \ldots \wedge C_n(x)$
concept negation	$\neg C$	$\neg C(x)$
universal quantification	$\forall P.C$	$\forall y.(P(x,y) \rightarrow C(y))$
existential quantification	$\exists P.C$	$\exists y.(P(x,y) \wedge C(y))$

Algorithm 1. Representation of the Intentional Relations Σ_{GM}

1: Input: Goal model $GM = (\mathcal{G}, \mathcal{C}, \mathcal{D})$
2: **if** $(G, IOR, \{G_1, \ldots, G_n\}) \in \mathcal{D}$ **then**
3: $\quad Rel_{G_i} \equiv \bigsqcup_{j=1,\ldots,n} \exists requires.G_j$ $(for\ i = 1, \ldots, n)$
4: **end if**
5: **if** $(G, AND, \{G_1, \ldots, G_n\}) \in \mathcal{D}$ **then**
6: $\quad Rel_{G_i} \equiv \bigsqcap_{j=1,\ldots,n} \exists requires.G_j$ $(for\ i = 1, \ldots, n)$
7: **end if**
8: **if** $(G, XOR, \{G_1, \ldots, G_n\}) \in \mathcal{D}$ **then**
9: $\quad Rel_{G_i} \equiv (\bigsqcup_{G' \in \{G_1,\ldots,G_n\}} \exists requires.G'$
$\quad \sqcap \neg(\bigsqcup_{G'',G''' \in \{G_1,\ldots,G_n\}} (\exists requires.G'' \sqcap \exists requires.G'''))$
10: **end if**
11: **for all** $(G', \dot{+}, G) \in \mathcal{C}$ **do**
12: $\quad Rel_G := Rel_G \sqcap \exists requires.G'$
13: **end for**
14: **for all** $(G', \dot{-}, G) \in \mathcal{C}$ **do**
15: $\quad Rel_G := Rel_G \sqcap \neg\exists requires.G'$
16: **end for**

previous case, we introduce relation concepts Rel_{G_i} to capture conjunction for all members of the decomposition.

The representation of exclusive decompositions is straightforward (lines 8–10). The concept union describes that at least one intentional element G' has to be satisfied in order to satisfy G, like in inclusive or decompositions. The second part of the expression excludes the case that more than one intention (G'' and G''') is satisfied. This is expressed by the concept negation ($\neg$) that precedes the second part of the concept expression. An intention can only be the source of one decomposition, i.e., either IOR, XOR or AND.

Sufficient positive contributions (lines 11–13) specify that the fulfillment of G requires the fulfillment of G'. Thus, we add the expression $\exists requires.G'$ to the definition of the relation concept Rel_G. Sufficient negative contributions (lines 14–16) use concept negation in order to represent that the fulfillment of G can not be achieved if G' is fulfilled. A task might be involved in multiple (positive and negative) contributions simultaneously.

Algorithm 2. Representation of the Workflow Patterns Σ_{PM}

Input: Materialized process model $PM = (\mathcal{V}, \mathcal{E}, \mathcal{F}, \mathcal{E}_A)$
for all $A \in \mathcal{A}$ $(\mathcal{A} \subseteq \mathcal{V})$ **do**
  $Rel_A \equiv \top$
end for
for all $E \in \mathcal{E}_A$ **do**
  if $\langle A_1, A_2 \rangle = E$ **then**
    $Rel_{A_1} := Rel_{A_1} \sqcap \exists requires.A_2$ and $Rel_{A_2} := Rel_{A_2} \sqcap \exists requires.A_1$
  end if
end for
for all $F \in \mathcal{F}$ **do**
  if $F = (and, and, \mathcal{B}) \vee F = (and, or, \mathcal{B}) \vee F = (and, xor, \mathcal{B}) \vee F = (and, disc, \mathcal{B})$ **then**
    $Rel_{A_i} := Rel_{A_i} \sqcap \bigsqcap_{B_j \in \mathcal{B}} \exists\ requires.(\bigsqcap_{A_k \in B_J} A_k)$
  end if
  if $F = (ior, ior, \mathcal{B}) \vee F = (ior, disc, \mathcal{B}) \vee F = (ior, xor, \mathcal{B})$ **then**
    $Rel_{A_i} := Rel_{A_i} \sqcap \bigsqcup_{B_j \in \mathcal{B}} \exists\ requires.(\bigsqcap_{A_k \in B_J} A_k)$
  end if
  if $F = (xor, xor, \mathcal{B})$ **then**
    $Rel_{A_i} := Rel_{A_i} \sqcap (\bigsqcup_{B_j \in \mathcal{B}} \exists\ requires.(\bigsqcap_{A_k \in B_J} A_k)) \sqcap \neg(\bigsqcap_{B_j \in \mathcal{B}} \exists\ requires.(\bigsqcap_{A_k \in B_J} A_k))$
  end if
end for

$$Rel_{ApproveOrder} \equiv \exists\ requires.ApproveOrder \sqcap \exists requires.CustomerProfileRetrieve \quad (1)$$

$$Rel_{ElectronicPayment} \equiv \exists requires.ElectronicPayment \sqcup \exists requires.InPersonPayment \quad (2)$$

Axiom 1 describes an AND-relation of the sibling intentional elements *ApproveOrder* and *CustomerProfileRetrieve*. The intentional relation of *CustomerProfileRetrieve* is defined equally. The AND-relation is expressed by the concept intersection ($\sqcap$) of the concept expressions $\exists\ requires.ApproveOrder$ and $\exists requires.CustomerProfileRetrieve$.

An inclusive OR-relation between *ElectronicPayment* or *InPersonPayment* is exemplified in Axiom 2, in which the fulfillment relationship of both tasks is reflected, while both tasks are inclusive siblings within an OR-decomposition. The relation of intention *InPersonPayment* is defined equally.

Workflow Relations. We represent business process models in terms of control flow relations between activities (cf. Sect. 2). Algorithm 2 describes how the corresponding knowledge base Σ_{PM} is built.

There might be an overlapping of activity relations. For instance, the activity Cash in Fig. 1(b) is part of an exclusive branching fragment (internal fragment in Fig. 1(b)) and also within a parallel branching fragment, i.e., the activity Cash is conjunctively and exclusively related to other activities. Accordingly, we build relations of activities (Rel_A) as a conjunction (intersection in DL) of activities

from the different control flow patterns. Initially, each relation concept Rel_A is defined as equivalent to the universal concept $\top$ (line 3).

In lines 5–9, the sequential control flow relations (in both directions) of an activity A are covered by restricting the concept Rel_A. Since gateways do not realize goals, they are transparent in the representation (cf. Def. 3). Afterwards, relations within fragments $\mathcal{F}$ are considered, whereby only those fragments that start and end with a gateway are relevant. Each branch $B \in \mathcal{B}$ of a fragment $F \in \mathcal{F}$ is a set of activities. In lines 11–13, the algorithm restricts the concept definitions Rel_{A_i}, in which A_i are activities in parallel branches. We use an intersection between activity sets of sibling branches, indicating the conjunctive relationship between sibling activities in parallel branches. From a logical point of view, we treat different exit gateways (multiple merge, synchronization and discriminator) equally. Branching relations do not impose restrictions on activities within the same branch in a fragment (in contrast to the sequence pattern). Thus, we describe all activities of the same branch by a concept intersection, independent of the kind of branching.

Multi choices are treated in lines 14–16, including synchronizing merge, simple merge and discriminator. Logically, activities of sibling branches are connected by a concept union. In case of exclusive branchings (lines 17–19), the concept definitions Rel_{A_i} contain a further restriction that allows only the execution of one branch. Like in goal models, this is expressed by a concept negation ($\neg$) in the DL concept definition. In both cases, activities of the same branch are treated like in the parallel case, i.e., they are represented by a concept intersection since IOR and XOR relations refer to activities of sibling branches, but not to activities of the same branch.

Axiom 3 depicts a part of the control flow relation of activity *CreditRateChecking* (cf. Fig. 1(b)). The activity is part of a choice fragment, i.e., either activity *CreditRateChecking* or *CustomerRecordsChecking* can be executed, or even both. This relation is represented by a concept union in the first line of the axiom. The second line of the axiom covers sequential relations of the activity *CreditRateChecking* to its predecessor (*CustomerIdentification*) and its successor (*ApplyDiscount*). Axiom 4 describes the relation of activity *FraudDetection*, as member of a parallel branch, in which activity *ApplyDiscount* and the internal fragment with activities *DebitCard*, *CreditCard* and *Cash*. The relation concept also covers predecessor (*SelectPaymentMethod*) and successor (*BuildAndPackageOrder*) activities.

$$
\begin{aligned}
Rel_{CreditRateChecking} \equiv\ & (\exists\ requires.CreditRateChecking \\
& \sqcup \exists requires.CustomerRecordsChecking) \\
& \sqcap\ \exists requires.CustomerIdentification \sqcap \exists requires.ApproveOrder \quad (3) \\
Rel_{FraudDetection} \equiv\ & (\exists\ requires.FraudDetection\ \sqcap\ \exists\ requires.ApplyDiscount \\
& \sqcap \exists\ requires(DebitCard \sqcap CreditCard \sqcap Cash)) \\
& \sqcap \exists\ requires.SelectPaymentMethod \\
& \sqcap \exists\ requires.BuildAndPackageOrder \quad (4)
\end{aligned}
$$

Mapping between Goals and Activities. Besides intentional relations and workflow patterns, we have to represent the realization of tasks by the corresponding activities in terms of mappings in the knowledge base Σ_M. A mapping is described as a concept equivalence in the knowledge base. If there is a mapping $m(G, A)$ from a task G to an activity A, we represent the mapping by an axiom $A \equiv G$. In both models, we use the same role *requires* in order to allow for a comparison of relations of both models.

5 Validation of Realization Equivalence

Mappings describe the realization of an intentional element by an activity. As a consequence, it is expected that the relations of an intentional element and its corresponding activity are not contradicting.

For a given mapping $m(G, A)$, we compare the corresponding workflow patterns WF of activity A and the intentional relations IR of intentional element G in order to test whether they are realization equivalent or not. Their relations are represented by concepts Rel_G and Rel_A. In case they are not realization equivalent, we want to know whether the reason is a strong inconsistency or a potential inconsistency, regarding to the distinction of Table 1 (Sect. 3).

From a logical point of view, concepts Rel_G and Rel_A represent formulas. The three different cases of inconsistencies and realization equivalence are reflected by the concepts Rel_G and Rel_A as follows: (i) A strong inconsistency means that there can not be any execution combination of activities that fulfills the intentional relations of the corresponding intentional elements. In the DL sense, the intersection of both concepts $Rel_G \sqcap Rel_A$ is unsatisfiable, i.e., the intersection $Rel_G \sqcap Rel_A$ cannot have a common individual. (ii) A potential inconsistency indicates that there might be execution combinations of activities, in which the corresponding intentional relations are not fulfilled. In this case, the intersection $Rel_G \sqcap Rel_A$ is satisfiable, i.e., there are common individuals of both concepts. (iii) The intentional relations of G and the workflow patterns of activity A are realization equivalent if all execution combinations that involve activity A lead to a fulfillment of the intentional relations of G. This is true if the subsumption $Rel_A \sqsubseteq Rel_G$ holds. Logically, this means that Rel_A implies Rel_G.

In order to check these three different cases, we introduce the following validation concepts. The concept $Valid^{\checkmark}$ is defined as $\neg Rel_A \sqcup Rel_G$ to encode the subsumption test $Rel_A \sqsubseteq Rel_G$. Thus, Rel_A is subsumed by Rel_G if $Valid^{\checkmark} \equiv \neg Rel_A \sqcup Rel_G$ is equivalent to the universal concept $\top$. This indicates the realization equivalence. A concept $Valid^{\pm}$ is defined as the intersection $Rel_A \sqcap Rel_G$ to test whether there is a potential or a strong inconsistency, i.e., if $Valid^{\pm}$ is satisfiable there is a potential inconsistency, otherwise a strong inconsistency. Formally, the knowledge base Σ is obtained as described in Def. 7.

Definition 7 (Final Knowledge Base Σ). *The knowledge base $\Sigma := \Sigma_{GM} \cup \Sigma_{PM} \cup \Sigma_M$ is extended as follows: For each mapping $m(G, A)$*

from an intentional element G to an activity A, which is represented in Σ_M by an axiom $G \equiv A$, we insert the following axioms into Σ:
(1) $Valid^{\checkmark}_{A,G} \equiv \neg Rel_A \sqcup Rel_G$ *and (2)* $Valid^{\pm}_{A,G} \equiv Rel_A \sqcap Rel_G$

Given the final knowledge base, we get the validation result *en passant*. Classifying the validation concepts $Valid^{\checkmark}_{A,G}$ and $Valid^{\pm}_{A,G}$ of the knowledge base Σ for each mapping $m(G, A)$ leads to the following results:

1. If $Valid^{\checkmark}_{A,G}$ is classified equal to the universal concept $\top$, we can guarantee the realization equivalence of the workflow patterns over activity A and the intentional relations over intentional element G.
2. In the other case, there is either a strong inconsistency or a potential inconsistency. This is indicated by the classification of the concept $Valid^{\pm}_{A,G}$. If $Valid^{\pm}_{A,G}$ is classified as a subconcept of the bottom concept $\bot$ (i.e., $Valid^{\pm}_{A,G} \sqsubseteq \bot$), there is a strong inconsistency, otherwise (i.e., $Valid^{\pm}_{A,G} \not\sqsubseteq \bot$) there is a potential inconsistency.

6 Proof-of Concept and Discussion

We conduct an evaluation by providing a proof-of concept, in which transformations of workflow patterns from BPMN models and intentional relations from GRL goal models into an OWL DL knowledge base are implemented.

Setting and Data Set. To calibrate our approach, we have used 20 goal models and 20 BPMN models, derived in different variants from the e-store case study [22]. The goal models have on average 24 goals and about 60 % of the goals are tasks, which can be realized by activities. The average size of the business process models is about 21 activities, the maximum number is 34 activities and the maximum depth of nested fragments is 4. In each setting, about 80 % of the activities are mapped to at least one task of the goal model.

Given a goal model, a business process model and mappings between both models, our tool creates the knowledge base Σ in an average time of 2480 msec. The DL expressiveness of Σ is $\mathcal{ALC}$. After reasoning on the knowledge base Σ, our tool produces a list of realization equivalent mappings ($Valid^{\checkmark} \equiv \top$) and a list of strong violations $Valid^{\pm} \sqsubseteq \bot$, the remaining are known as potential violations (i.e., $Valid^{\pm} \not\sqsubseteq \bot$). The reasoning time for the classification is on average 3480 msec. The ontology creation is implemented with the OWL-API[4]. For reasoning, we used the Pellet reasoner[5]. Our test system is a Notebook with an Intel Core 2 Duo 8700 CPU (2.5 GHz, 4 MB cache and 2GB DDR2 RAM).

Validation Exemplified. We demonstrate the validation for an excerpt of Fig. 1. Consider the strong inconsistency for the activities *SendBill* and *Shipment*. They are siblings in exclusive branches of the same fragment. In

[4] OWL-API site: `http://owlapi.sourceforge.net/`
[5] Pellet reasoner site: `http://clarkparsia.com/pellet/`

Σ_{PM}, there are the definitions of this relation for both activities. An excerpt of the relation definition of activity *SendBill* is shown in Axiom 5.

$$Rel_{SendBill} \equiv (\exists\ requires.SendBill \sqcup \exists\ requires.Shipment) \sqcap \neg(\exists\ requires.SendBill \sqcap \exists\ requires.Shipping) \quad (5)$$

$$Rel_{Bill} \equiv \exists\ requires.ShipOrder \sqcap \exists\ requires.Bill \quad (6)$$

Since *SendBill* is mapped to the intentional element Bill (BL), *SendBill* is defined as equivalent to the concept *Bill* and *Shipping* is equivalent to *ShipOrder* (mapping knowledge base Σ_M). From the goal model, there is a relation definition of *Bill* in Σ_{GM} as depicted in Axiom 6.

The validation concept $Valid^{\checkmark} \equiv \neg Rel_{SendBill} \sqcup Rel_{Bill}$ is classified by the reasoner as different from the universal concept $\top$, i.e., we know that the realization equivalence does not hold between *SendBill* and *Bill*. The other validation concept $Valid^{\pm} \equiv Rel_{SendBill} \sqcap Rel_{Bill}$ is classified equivalent to the bottom concept $\bot$, i.e., indicating a strong inconsistency. The same holds for *Shipment*.

Lessons Learned and Limitations. As mentioned in Sect. 3, we restrict mappings to atomic tasks in the goal model since they operationalize stakeholders' goals, and therefore, they can be directly realized by activities in a process. Discussions whether hard and especially soft goals could be mapped to activities are outside of the scope of this paper. The proposed modeling framework is based on these assumptions. However, the modeling and validation approach is quite generic and DL is expressive enough to incorporate further aspects like mappings of other intentional elements like soft goals to activities.

The purpose of the validation is to detect inconsistencies between mapped elements with respect to their relationships, i.e., intentional relations and relations of activities. This is based on a comparison how these elements are logically related to each other, e.g., whether their relations (or constraints) coincide or contradict each other. However, an analysis whether or to which degree the execution of an activity satisfies a particular goal from a qualitative perspective leads to further interesting research questions.

7 Related Work

The first group of related work considers relations between goal models and business process models, but not validation as it is done in our work. Transforming goal models into business process model has been one of the major research areas in business process management. Lapouchnian et al. [23] use goal models for the configuration of business processes. Goal models are annotated with control flow information, they are transformed into BPEL processes. Similarly, Decreus and Poels [24] annotate goal-oriented models in the so-called B-SCP framework with control flow information and transform them into BPMN skeletons. Furthermore, Frankova et al. [25] transform SI*/Secure Tropos models into skeletons

of process models in BPMN, from which they generate executable processes in BPEL. On the other hand, Santos et al. [4] derive goal models from existing process models. Such goal models are then used to control variability and configuration of processes. Finally, Koliadis et al. [2] annotate activities of process with effects, whereby these effects serve for a comparison of a process with goals, i.e., effects of activities are compared with goal fulfillments. The basic principle is to reflect changes from an $i^\star$ model to a BPMN model and vice versa.

The second group of related research is more related to our work, where some kind of verification between requirements and business process models is investigated. In this line of related research, Kazhamiakin et al. [26] propose a methodology that provides a set of high-level mapping rules to produce process models in BPEL from Tropos models. In order to enable requirements driven verification of process models, they employ the formal Tropos language and temporal constraints. While their work and tools support several types of formal analyses, they do not focus on validation of process models with respect to the satisfaction of goal models, as it is done in our work. Soffer and Wand [27,28] propose a generic theory-based process modeling framework based on Bunge's ontology for a goal-driven analysis of process models to check goal reachability in process models. They define a goal as stable state, which indicates the termination of the process. Their framework defines a set assumptions and parameters (based on goal relations and workflow relations), which are used to check if a set of workflow patterns ensure that processes can always reach to their goals. Hence, using these parameters it is possible to identify set of valid and invalid design decision with respect to business goals. They also suggest appropriate redesign actions to eliminate these invalid designs. Our approach follows the similar objective, but we use DL based reasoning to identify inconsistencies automatically.

Liaskos et al. [29] also introduce an approach for goal based customization of workflows. A family of processes is extended with the notation for partial temporal ordering of goals. A particular member of the family is specified by constraints with the means of linear temporal logic operators. However, this approach does not take into consideration business process logic embedded in the realization of business processes, which is the focus of our validation. La Rosa et al. [30] propose a questionnaire based customization of configurable business processes represented in C-YAWL [31]. They use a Petri-net based reasoner [32] to preserve the correctness during process configuration. Variability patterns of La Rosa et al.'s work are narrower than patterns introduced in this work.

8 Conclusion

In this paper, we have presented a novel approach for handling inconsistencies resulting from mapping goal models and business process models. By automatically identifying inconsistencies, we are able to detect executable processes that did not meet user requirements or lead to undesired executions. Additionally, we allow for goal models and process models to evolve independently. Our contribution extends the body of knowledge in the field by considering these mapping as first-class citizens along with goal models and business process models. We plan

to extend this approach in combination with our existing work on configuration of business process families [33] to provide a complete solution for software product lines that use goal models, feature models and process models as main artifacts.

References

1. Dumas, M., Recker, J., Weske, M.: Management and engineering of process-aware information systems: Introduction to the special issue. Information Systems 37(2), 77–79 (2012)
2. Koliadis, G., Vranesevic, A., Bhuiyan, M.A., Krishna, A., Ghose, A.K.: Combining i^* and BPMN for Business Process Model Lifecycle Management. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 416–427. Springer, Heidelberg (2006)
3. Kueng, P., Kawalek, P.: Goal-based Business Process Models: Creation and evaluation. Business Process Management Journal, 17–38 (1997)
4. Santos, E., Castro, J., Sánchez, J., Pastor, O.: A Goal-Oriented Approach for Variability in BPMN. In: Workshop em Engenharia de Requisitos, WER (2010)
5. Andersson, B., Bergholtz, M., Edirisuriya, A., Ilayperuma, T., Johannesson, P.: A Declarative Foundation of Process Models. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 233–247. Springer, Heidelberg (2005)
6. Bergholtz, M., Jayaweera, P., Johannesson, P., Wohed, P.: A Pattern and Dependency Based Approach to the Design of Process Models. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 724–739. Springer, Heidelberg (2004)
7. Russel, N., ter Hofstede, A., van der Aalst, W., Mulyar, N.: Workflow Control-Flow Patterns: A Revised View. Technical report, BPM Center Report BPM-06-22, BPMcenter.org (2006)
8. Yu, E., Giorgin, P., Maiden, N., Mylopoulos, J. (eds.): Social Modeling for Requirements Engineering. MIT (2011)
9. Giorgini, P., Mylopoulos, J., Sebastiani, R.: Goal-oriented requirements analysis and reasoning in the TROPOS methodology. Engineering Applications of Artificial Intelligence 18, 159–171 (2005)
10. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering, 1st edn. Springer (1999)
11. van Lamsweerde, A.: Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley (2009)
12. ITU-T: Recommendation Z.151 (09/08): User Requirements Notation (URN) Language Definition (2008)
13. Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., Yu, E.: Evaluating Goal Models within the Goal-Oriented Requirement Language. International Journal on Intelligent Systems 25, 841–877 (2010)
14. Ouyang, C., Dumas, M., Aalst, W.M.P.V.D., Hofstede, A.H.M.T., Mendling, J.: From business process models to process-oriented software systems. ACM Trans. Softw. Eng. Methodol. 19, 2:1–2:37 (2009)
15. Feiler, P.H., Humphrey, W.S.: Software process development and enactment: concepts and definitions. In: Second International Conference on the Software Process, Continuous Software Process Improvement, pp. 28–40 (February 1993)
16. Johnson, R., Pearson, D., Pingali, K.: The Program Structure Tree: Computing Control Regions in Linear Time. In: PLDI, pp. 171–185 (1994)

17. Küster, J.M., Gerth, C., Förster, A., Engels, G.: Detecting and Resolving Process Model Differences in the Absence of a Change Log. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 244–260. Springer, Heidelberg (2008)
18. Vanhatalo, J., Völzer, H., Leymann, F.: Faster and More Focused Control-Flow Analysis for Business Process Models Through SESE Decomposition. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 43–55. Springer, Heidelberg (2007)
19. Kiepuszewski, B., ter Hofstede, A.H.M., Bussler, C.J.: On Structured Workflow Modelling. In: Wangler, B., Bergman, L.D. (eds.) CAiSE 2000. LNCS, vol. 1789, pp. 431–445. Springer, Heidelberg (2000)
20. van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow Patterns. In: Distributed and Parallel Databases (2003)
21. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook, 2nd edn. Cambridge University Press (2007)
22. Lau, S.Q.: Domain Analysis of E-Commerce Systems Using Feature-Based Model Templates. Master's thesis, University of Waterloo, Waterloo (2006)
23. Lapouchnian, A., Yu, Y., Mylopoulos, J.: Requirements-Driven Design and Configuration Management of Business Processes. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 246–261. Springer, Heidelberg (2007)
24. Decreus, K., Poels, G.: A Goal-Oriented Requirements Engineering Method for Business Processes. In: Soffer, P., Proper, E. (eds.) CAiSE Forum 2010. LNBIP, vol. 72, pp. 29–43. Springer, Heidelberg (2011)
25. Frankova, G., Frankova, G., Massacci, F., Massacci, F., Seguran, M., Seguran, M.: From early requirements analysis towards secure workflows. Technical Report TR DIT-07-036, University of Trento (2007)
26. Kazhamiakin, R., Pistore, M., Roveri, M.: A framework for integrating business processes and business requirements. In: Proc. IEEE DOC 2004 Conf., pp. 9–20 (2004)
27. Soffer, P., Wand, Y.: Goal-Driven Analysis of Process Model Validity. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 521–535. Springer, Heidelberg (2004)
28. Soffer, P., Wand, Y., Kaner, M.: Semantic Analysis of Flow Patterns in Business Process Modeling. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 400–407. Springer, Heidelberg (2007)
29. Liaskos, S., Litoiu, M., Jungblut, M.D., Mylopoulos, J.: Goal-Based Behavioral Customization of Information Systems. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 77–92. Springer, Heidelberg (2011)
30. La Rosa, M., van der Aalst, W., Dumas, M., ter Hofstede, A.: Questionnaire-based Variability Modeling for System Configuration. Software and Systems Modeling 8, 251–274 (2009)
31. Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H., Rosa, M.L.: Configurable workflow models. International Journal on Cooperative Information Systems 17(2), 177–221 (2008)
32. van der Aalst, W., Dumas, M., Gottschalk, F., ter Hofstede, A., La Rosa, M., Mendling, J.: Preserving correctness during business process model configuration. Formal Aspects of Computing 22, 459–482 (2010)
33. Gröner, G., Wende, C., Bošković, M., Silva Parreiras, F., Walter, T., Heidenreich, F., Gašević, D., Staab, S.: Validation of Families of Business Processes. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 551–565. Springer, Heidelberg (2011)