Recommendation Patterns for Business Process Imperative Modeling

Mateus Barcellos Costa
Software Eng. and e-Commerce Lab. – SEEC
Informatics - Federal I. of Espírito Santo
Serra – ES, Brazil
mcosta@ifes.edu.br

Dalila Tamzalit
LS2N CNRS UMR 6004
Université de Nantes
Nantes, France
dalila.tamzalit@univ-nantes.fr

ABSTRACT

With the wide acceptance of the Imperative Business Process Modeling, supporting this activity has becoming increasingly important. In this sense, the difficulties faced by modelers while mapping business concerns into correct process models is a key issue. This paper presents the idea of Imperative Modeling Recommendation Patterns in order to tackle this issue. Recommendation Patterns aim at supporting modelers to decide between modeling structure options while preserving both process and modeling concerns. By doing so, these patterns enable the achievement of correct modeling solutions without, however, forcing a unique solution. The application of the Recommendation Patterns is discussed in the context of a methodology for Business Process Imperative Modeling support.

CCS Concepts

•**Applied computing → Business process modeling;**

Keywords

Business Processes; Modeling; Recommendation Methodology

1. INTRODUCTION

Imperative Business Process Modeling has been widely used to document, so as to support system automation, in different application domains, ranging from typical scenarios, e.g. Office Information Systems [10], to novel applications, e.g. Sensor Networks design [7]. Models resulting from the application of this paradigm usually consist of visual graph-based descriptions whereby the essence of process related elements is understood, captured and synthesized [22].

In order to be successful, Imperative Modeling may require methodological support [17]. In this sense, a general question that one might ask is what hinders the modeler to use correct model solutions and what makes him to violate modeling guidelines. In particular, it can be observed a cognitive hurdle [1] when the business concerns that govern the process [11] lead to more than one modeling solution. In this case, even simple situations require a decisional process to create process models accordingly to process domain issues. For example, for a process having three activities $\boldsymbol{a}$,$\boldsymbol{b}$ and $\boldsymbol{c}$, and one execution dependence constraint saying that $\boldsymbol{b}$ must not occur before $\boldsymbol{a}$, it can be verified that there are at least five valid imperative model solutions. The modeling process can easily become more complicated, i.e., with more modeling options, by adding a few more activities and constraints. Moreover, the decisional process may depend on several relevant process quality concerns [4] in order to evaluate possible solutions.

To tackle this issue, the so-called Imperative Modeling Recommendation Patterns are discussed in this paper. Recommendation Patterns can be applied to obtain suggestions of model fragments at a certain modeling stage, for a given set of process concerns. Recommendation Patterns can, therefore, support modelers to decide between modeling structure options by considering both process and modeling concerns. In the proposed solution, process concerns are described by using the concept of *Situation* [2] while modeling concerns are captured with Business Process Modeling and Notation (BPMN) [18] syntactical valid model fragments (recommendations).

A preliminary catalog with five recommendation patterns is presented and the basis for an application methodology is also introduced. The benefits of Recommendation Patterns usage include the compliance of process models with a process conceptual view given by a set of situations. In addition, it can be showed, by the methodology application, that modelers can be driven through the modeling solution space without, however, being forced into a single solution. Besides these benefits, the methodology application has suggested that the adoption of modeling guidelines could also be supported.

The remainder of this paper is organized as follows: Section 2 discusses Business process Modeling Support. The central problem tackled by this work and its proposed approach are also discussed in Section 2. Section 3 introduces Recommendation Patterns and discusses an application methodology. In Section 4, a process modelling application example is used to illustrate the provided methodology. Section 5 discusses Recommendation Patterns limitations and extensibility aspects and Section 6 concludes the paper.

SAC 2017, April 03-07, 2017, Marrakech, Morocco
© 2017 ACM. ISBN 978-1-4503-4486-9/17/04...$15.00
DOI: http://dx.doi.org/10.1145/3019612.3019619

2. IMPERATIVE BUSINESS PROCESS MODELING SUPPORT

Numerous works have set up modeling guidelines and tools to support process modeling. Relevant work includes the so-called Workflow Patterns [24] which enable, among other applications, the identification of recurrent problems frequently found in workflow design and generic design solutions to such problems. As such, modelers can use workflow patterns in order to get support to build imperative business models. However, as a specialized type of design pattern, the use of workflow patterns is intended to provide solution for problems which are already specified as the design option. For example, the sequence pattern $a \rightarrow b$ (a followed by b) is the design option for the conceptual dependence constraint on b towards a, but it may not be the unique design solution for such constraint.

The design of process models has been also supported by modeling guidelines [5, 17]. Mendling et al [17], provides seven modeling guidelines based on an empirical study of how people create good process models. These guidelines form a pragmatic set of modeling rules which can be directly applied in order to draw good process models. Interpreting and following these rules, however, remain susceptible to the modeling context.

Some recent works have also considered the assistance of recommender systems capable of suggesting model fragments and properties. Initiatives in this direction have focused on finding already accepted solutions, based on, for example, reference models, model repositories and execution historical data. Recommendations include, for example, suggestions of model fragments which are deemed appropriate with respect to process standards [9, 25] or proven solutions [23, 14, 16], and resource and service allocation solutions for process instantiation [8]. Although recommendation systems can provide useful insights with respect to business process problems, the adoption of recommendations from different contexts may require adaptation and integration with particular model assumptions. Moreover, specific process modeling problems can be difficult to be mapped into already used solutions and may still require specific support.

2.1 Problem Statement

Modeling support initiatives, as the ones discussed above, aim at, above all, providing good engineering practice so they can be regarded as "golden rules" for this task [6]. Even though these practices have effectively contributed to Process Modeling, it is also a key issue to consider the difficulties faced by modelers to overcome problems related to both modeler and process perspectives.

From the modeler's perspective, this question can be summed up as the idea of changing the focus of supporting solutions from *What improves modeling?* to *What hinders modeling?*. In this sense, Mendling et al [17] observed that users hardly get any support to create good process models. In additional, they consider the low level of modeling competence that many casual modelers have as a recurrent problem in Process Modeling.

From the Process perspective, it is assumed that imperative process models are the resulting specification of a Process Analysis activity [12]. However they are not actually a conceptual process view but a model solution to that, which imposes a specific behavior. This assumption can be sustained by the example given in Section 1. In this example, the execution dependence between a and b is a conceptual process constraint which requires that some situation (a occurs before b), should occur at the process execution. In order to satisfy this constraint, it is sufficient that a occurs before b and not just before. Thus, various model alternatives are allowed. In this sense, it seems that a more realistic first definition, able to capture the business process conceptual view, can be obtained by descriptions of ***what*** situations should occur at process execution rather than by descriptions about ***how*** these situations should occur.

Considering such perspectives, a particular problem which is discussed in this paper, is the decisional process that takes place when a process modeling problem enables several modeling options. Thus, more specifically, the problem approached in this paper can be stated as:

How to support modelers to decide among modeling structure options while taking care of process concerns and avoiding the generation of error-prone models.

2.2 Proposed Approach

This work deals with a perspective on modeling support which explores the universe of possibilities enabled by the business process conceptual view. In this sense, the objective of Recommendation Patterns is, therefore, to provide imperative modeling options which are in conformity with this view. The term recommendation is used to emphasize that the patterns are able to give all possible modeling options provided by the process conceptual view.

The basis from where recommendations derive is the process behavior determined by a *Situation*. As discussed by Adi and Etzion [2], a situation can be regarded as a description of a relevant happening that does not occur explicitly but can be inferred by viewing the world's state. For example, a dependence on activity b towards a can not be taken directly from the concrete reality but from the observation in process execution that a is always performed before b.

In the context of Complex Events a situation defines the set of events that need to be evaluated, and conditions that must be satisfied when they occur. Indeed, a situation can be fulfilled by different complex (inferred) events acting as an abstract collection of event patters which describe these inferred events. In this work, a situation is used to describe the business process expected behavior. By doing so, a model that causes a situation is said to be in compliance with the situation itself.

A model causes a situation if its behavior matches up with at least one pattern it represents. Therefore, in the modeling context, given a situation, the modeler can opt to one of the patterns it defines. Since each pattern can be provided by a different imperative model structure, more than one modeling option can be obtained.

Situations can be described in various ways and can be of different natures. They can include, for instance, logic and temporal relations or even domain specific business rules. To formulate a complete set of situations is out of the scope of this work. However, in order to demonstrate the idea of Recommendation Patterns and its application, it is considered a set of situations which are recurrent and whose support can be found on business process declarative languages, e.g, Declare [19] and SBVR [11].

In order to characterize process as well as situation behavior, it is used the concept of **process state**. A process state records the level of advance of a process execution in terms of what active flow objects (activities or events) were performed in time. The process behavior, being the set of execution flows expected for the process, can be described in terms of transitions among the process states that are reached during the course of these flows. The behavior of a situation also determines the so-called mandatory states. A **mandatory state** is a process state whose reach must be granted by some execution flow of a process model, so that this model becomes in conformity with the situation. The mandatory states come from situation semantics so that its definition is a subject of interpretation or formal semantics depending on the situation case. The concept of mandatory state enables us to define **validity** of process model fragment. A model fragment is said valid with respect to a set of situations if all mandatory states determined by them are reachable.

To represent a situation behavior, Hasse diagrams [3] are used. A Hasse diagram is a kind of graph used to represent finite partially ordered sets in a form that indicates some dynamic property, e.g. insertion order. When representing process behavior, process states are mapped to vertices and each edge represents a flow object execution which is possible from its respective incoming vertex. Hasse diagrams are used to illustrate situations' behavior as it can be seen in Table 1. These diagrams indicate valid execution flows and what mandatory states, (marked with a t), must be granted to be reached from a given start state (marked with a s). Moreover, the diagram also gives the set of execution options to leave from certain process state. That is, for a state (a vertex in diagram) with two outcome edges A and B, it is true that this state can be left by executing the flow objects A or B.

3. RECOMMENDATION PATTERNS

A recommendation pattern maps a situation into a set of recommendations. This section highlights this mapping by discussing these concepts and the and the process of obtaining recommendations. In order to clarify Recommendation Patterns application aspects, a modelling support methodology is also discussed.

3.1 Situations

In the context of the Recommendation Patterns, a situation ($\mathscr{E}$) is described by a relation type over active flow objects. An **active flow object** is a process element representing something that happens (event) or work to be performed (activity) [21]. These elements belong to the process **Domain** ($\mathscr{D}$) , that corresponds to the set of active flow objects of a business process. The situations considered in the recommendation patterns presented in this work are presented below.

Independence. A set of active flow objects with no execution constraints. Any execution flow involving all objects of a independent situation is valid. The $*$ symbol is used to represent Independence. E.g. $*(I)$, indicates a situation where the active flow objects of the I set are independent from each other.

Dependence. A set of active flow objects with a temporal execution dependence among them. Dependence can be of two types:

Strict Dependence. In a strict dependence relation, if the flow object b depends on the flow object a, then b can be executed in a flow only and only if a has been executed before.

Circumstantial Dependence. In a circumstantial dependence relation, if the flow object b depends on the flow object a, then b can be executed in a flow where a was executed before or, in a flow where a is not executed any time.

The $\lhd$ symbol is used to represent dependence. E.g., the expression $ds \lhd es$, indicates a dependence situation where the ds (**depender set**) is the set of active flow objects that depends on the set es (**dependee set**). In case of Circumstantial dependence, the symbol $\unlhd$ is used.

Non-coexistence. A set of flow objects with a non-coexistence relation at the same execution flow. If there is a non-coexistence relation between the flow objects sets $\{a\}$ and $\{b\}$, any execution flow executing both a and b is forbidden. The symbol $\otimes$ is used to represent non-coexistence situations. E.g., the expression ($ns_1 \otimes ns_2$) indicates a non-coexistence between the active flow objects set ns_1 and ns_2.

Union. A set of flow objects with a union relation at the same execution flow. A union relation between the flow objects set $\{a\}$ and $\{b\}$ requires a process model with a flow executing a, a flow executing b and a flow executing both a and b. The symbol $\oplus$ is used to represent non-coexistence situations. E.g., the expression ($ns_1 \oplus ns_2$) indicates a non-coexistence between the active flow objects set ns_1 and ns_2.

The group of flow objects which are engaged in a non-coexistence or union situation is named ***Choice set***. Situations can also use the *or* logical connector ($\vee$) to relate active flow objects. For example, the dependence situation $c \lhd (a \vee b)$ indicates that c depends on a or b.

Since this work focused on presenting the key ideas surrounding Recommendation Patterns, cycles and parallelism were not addressed in this preliminary set of situations. In the case of parallelism, it should be clear that, when process dependence and non-coexistence situations allow, the produced recommendations may contain parallel execution flows. However, it is not part of this initial catalog a situation that allows to prescribe the parallel execution of flow objects. That is, there is not a situation able to describe cases where parallelism exists at the conceptual level. Such cases would be those in which two or more flow objects must be executed simultaneously in order to a process goal to be achieved. For example, in an industrial process where two metals need to be melted simultaneously to be further mixed. As such cases are less common and the use of parallelism is almost always motivated by optimization concerns, it was decided do not to approach parallel situations. However, as already discussed, for processes that can be represented without the conceptual prescription of parallel flows, parallel flows can be adopted if recommendations include them.

3.2 Recommendations

Imperative process models and recommendations are represented here considering a sub set of BPMN. The following constructs were considered: event and activities, which are both indistinctly as active flow objects, flow rows and the exclusive (xor), parallel (and) and inclusive (or) gateways. This BPMN subset is a modeling simplification similar to the one assumed by Polyvyanyy at al [20].

A **Recommendation** ($\mathscr{R}$) is an imperative model fragment which can be introduced into a model while preserving its validity. Taking in consideration the adopted simplified BPMN subset, the following structures are valid recommendations[1]:

Sequential: A single active flow object. Sequential recommendation is represented by the flow object identifier.

Parallel: A Parallel gateway connecting one or more active objects flow . Parallel recommendation will be represented by the "plus symbol" followed by the parallel flow objects sets separated by semicolon, e.g., $+(a; b)$.

Or: An OR gateway connecting one or more active flow objects. It is represented by the "o" symbol followed by the inclusive flow objects sets separated by a semicolon, e.g., $o(b; b)$.

Exclusive: An exclusive or gateway of connecting one or more active flow objects. It is represented by the "$\times$ symbol" followed by the exclusive flow objects sets separated by a semicolon, e.g., $\times(a; b)$.

3.3 Deriving Recommendations from Situations

A fundamental property of any valid recommendation is that it must not preclude any mandatory process state to be reachable by the model fragment resulting from its inclusion. To illustrate this property, let us take a look at the behavior of the Independence situation which is showed as a Hasse diagram in Table 1. In the independence situation diagram, it could be observed that any execution flow involving any ordered combination of A, B and C of any size, leaving the initial state s does not prevent the process from reaching the mandatory state t. In this analysis, it should be observed that the process execution does not reach more than one process state at the same time. Nevertheless, since incoming edges indicate independence among the active flow object they represent, any execution flow formed by any execution order of these objects is enabled.

Table 1 illustrates the five recommendation patterns comprising the situations discussed in Section 3.1 and their respective recommendations. The domain $\mathscr{D} = \{A, B, C\}$ is used to exemplify patterns. Each recommendation must be interpreted as an imperative model fragment which can be introduced in a model point having the reachable state s marked in the Hasse diagram.

The recommendations for the Independence situation – $*(\mathscr{D})$ – is given by the set of sequential recommendations correspondent to all elements of $\mathscr{D}$ plus the set of parallel recommendations correspondent to each subset of $\mathscr{D}$ with size greater than two. A parallel recommendation for each object with the empty flow is also a valid. It can be observed at the Hasse diagram that the mandatory state t is not precluded, by any of these recommendations, from being reached. In the remaining patters, A and B represent the active flow objects which are involved in the situation and C is used to represent active flow objects in the domain which are not involved in the specified situation (neutral objects).

The recommendation set for the Strict Dependence situation – $ds \lhd es$ – is given by the set of sequential recommendations correspondent to the dependee set ds (A), the set of sequential recommendations correspondent to the neutral objects (C) plus the parallel gateway among elements of the dependee set es and neutral objects ($+(A; C)$).

The recommendation set for the Circumstantial Dependence is given by the sequential flow involving the object (C), the parallel gateway between C and the empty flow and the XOR gateway between A and the empty flow ($\times(A;)$). This recommendation can be perceived from the situation semantics. Informally, circumstantial dependence is a situation where some dependee activity is facultative to reach some objective. An example of such situation is a process to send a product to a client with the activities, Pick-up at supplier (A), Ship product (B) and Invoice (C), where the Pick-up at supplier task is executed only if the product is out of stock. In this case, Ship product depends on Pick-up at supplier only if it occurs.

Looking at the Hasse diagram of circumstantial dependence, it can be noted that the sequence flow with A can not be recommended since the execution of A from s turns the state t_1 unreachable. The sequence flow B also can not be recommended since the execution of B from s turns state t_2 unreachable. The sequence flow C can be recommended since the execution of C from s reaches state 2 from where both t_1 and t_2 remain reachable. the recommendation ($\times(A;)$) is possible since execution of A from s reaches state 1 from where t_2 is reachable and the empty flow leaves the process at state s, from where t_1 is reachable.

The recommendation set for the Non-Coexistence situation – $\mathscr{E} = A \otimes B$ – is given by the sequential flow with C, the parallel gateway between C and the empty flow, and the XOR gateway $\times(A; B)$. These fragments do not prevent the process execution from reaching both the mandatory states t_1 and t_2. If, for example, a parallel gateway between A and C was introduced on the model, state t_1 is reached but t_2 becomes unreachable by the model. Finally, the recommendation set for the Union situation, $\mathscr{E} = A \oplus B$, is given by the sequential recommendation of C, the parallel gateway between C and the empty flow, and the OR gateway between A and B ($+(A; B)$). In this situation, although state t_3 is reachable by introducing a parallel gateway among A, B and C, if this structure was taken, states t_1 and t_2 would be unreachable.

Following a given recommendation does not mean fulfilling a situation. Actually, it is the recommendation chaining, driven by the methodology, that will do it. When a recommendation is followed, what is guaranteed is the fact that the mandatory states defined by the set of process situations remain reachable. Therefore, a situation may have to be considered more than once during the whole modeling process until it is satisfied. For example, let us consider the Circumstantial Dependence situation and the following of the recommendation $\times(A;)$. At this point the situation is not fully satisfied. Next, the situation is also considered but taking into account the reachable states which are granted by prior recommendations. For the next step, the only one valid recommendation is a XOR gateway convergence[2] since any other inclusion of B or C makes at least one mandatory state

[1]For the sake of space, an algebraic notation is used to represent BPMN graphical structures in recommendation descriptions.

[2]A join gateway.

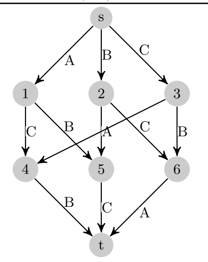
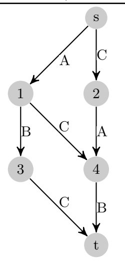
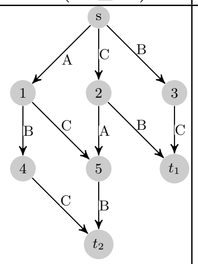
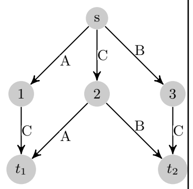
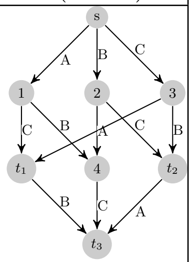
	Independence	Dependence	Circumstantial Dependence	Non-coexistence	Union
Situation	$\mathscr{D} = \{A, B, C\}$ $\mathscr{E} = *(\mathscr{D})$	$\mathscr{D} = \{A, B, C\}$ $\mathscr{E} = (B \lhd A)$	$\mathscr{D} = \{A, B, C\}$ $\mathscr{E} = (B \unlhd A)$	$\mathscr{D} = \{A, B, C\}$ $\mathscr{E} = (A \otimes B)$	$\mathscr{D} = \{A, B, C\}$ $\mathscr{E} = (A \oplus B)$
Behavior					
Recommendation	$A, B, C, +(A;), +(B;), +(C;), +(A;B), +(A;C), +(B;C), +(A;B;C)$	$A,\ C,$ $+(A;C)$	$C, \times(A;)$ $+(C;)$	$C,\ +(C;)$ $\times(A;B)$	$C, +(C;)$ $o(A;B)$

Table 1: Recommendation Patterns.

unreachable. Next, states *s* and *1* are now assuredly reachable and any recommendation from these starting points will keep all the mandatory states achievable. At this point, a parallel gateway between B and C could satisfy the situation.

3.4 Recommendation Methodology

In this section it is discussed a methodology for applying Recommendation Patterns. It is assumed that a conceptual process description, which makes explicit every mandatory situation, is provided. An imperative model is considered valid if it fulfils or does not violate any described situation.

Let us consider, as a running example, the drawing of a BPMN model to support a process with domain $\mathscr{D} = \{t_1,\ t_2,\ t_3,\ t_4\}$ and the situations $\mathscr{E}_1 = t_2 \lhd t_1$ and $\mathscr{E}_2 = t_3 \lhd t_1$. The process starts with an empty model. For the sake of readability, it is included an empty initial event in this model, so that this event is not included in the domain. At this stage there is one recommendation point which is marked with rp_1 in Figure 1. A **recommendation point** refers to a termination in a model branch where a model fragment can be inserted. Each recommendation point in turn, can have one or more reachable process states. A reachable process state of a recommendation point is a process state which can be reached when the process execution flows by the recommendation point. In order to represent process states, it is used a binary string where each bit represents flow objects individual states. E.g, the state 0000 indicates that no flow object was performed and, the state 0011 indicates that t_1 and t_2 flow objects have been performed.

The set of recommendations for each recommendation point should be determined separately, taking into account all of its reachable states. For rp_1 there is only the 0000 reachable state. In this case, the recommendation set, given by the Dependence pattern, for situation $\mathscr{E}_1$ is

$$\mathscr{R}_1 = \{t_1,\ t_3,\ t_4, +(t_1;t_3), +t(t_1;t_4),\ +(t_3;t_4), +(t_1;t_3;t_4)\}$$

and, for $\mathscr{E}_2$ is

$$\mathscr{R}_2 = \{t_1,\ t_2,\ t_4, +(t_1;t_2), +t(t_1;t_4), +(t_2;t_4), +(t_1;t_2;t_4)\}$$

The resulting recommendation set for rp_1 is given by $\mathscr{R} = \mathscr{R}_1 \cap \mathscr{R}_2$. They are the BPMN fragments showed in Figure 2.

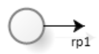


Figure 1: Initial BPMN model.

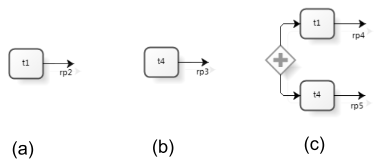


Figure 2: Recommendation options for rp_1.

Let us suppose that the modeler decides in favor of the recommendation (a). Again, there is only one recommendation point (rp_2) and the reachable state at rp_2 is 0001. The recommendation set for rp_2 is shown in Figure 3. Now, considering a decision about the recommendation (d), there will be two recommendation points: rp_9 *and* rp_{10}. For rp_9, the reachable states are 0011, when only t_1 and t_2 are done at this point and 0111, when t_1, t_2 and t_3 are already done. However, as there is no restriction to execute t_4, for both recommendation points we have the same recommendation set: the activity t_4 or the **empty recommendation** ($\emptyset$). The Empty recommendation is used to determine a gateway convergence. When there are empty recommendations in all recommendation points corresponding to gateway branches, a gateway convergence is included as a valid recommendation. The process ends when all activities are included in the model satisfying all situations.

The determination of the recommendation set for a recommendation point must consider the stricter reachable state. Let us suppose that the recommendation $+(t_1;t_4)$ (c) at rp_1 is chosen. In this case the recommendation point after t_1 would have the reachable states 1001 and 0001 and the recommendation point after t_4 would have the reachable states 1000 and 1001. In this case, the recommendation after t_1 would consider fragments with $t2$ and t_3 but, after t_4, it should consider only the empty recommendation since, at the reachable state 1000, the dependence constraints described by $\mathscr{E}_1$ and $\mathscr{E}_2$ were not already satisfied.

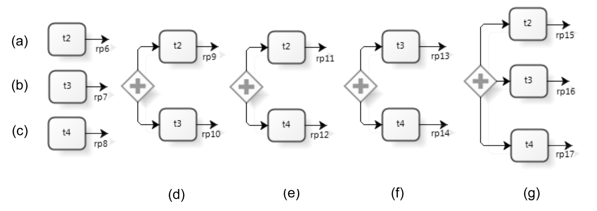


Figure 3: Recommendation options for rp_2

4. APPLICATION EXAMPLE

The methodology discussed in Section 3.4 was applied to the same example used by Mendling et al [17] to discuss modeling guidelines. This example describes the following procedure for handling complaints:

Handling Complaint process. *A new case is opened if a new complaint is received – be it as a phone call, as a personal contact, or as a letter. In some situations, the complaint must be referred, either internally or externally. Internal referrals have to be put on the incident agenda while external referrals require a confirmation. In both cases the referral is archived in parallel. Finally, the complainant is informed. If no referral is required, a complaint analysis is conducted. Later, the complaint is archived and the complainant is contacted with an optional follow up.*

The domain and the situations for the Handling Complaint process are shown respectively in Table 2 and Table 3. The flow objects in the domain and the situations were captured strictly from the textual description.

Figure 4 shows the BPMN model for the Handling Complaint process which was obtained by following the recommendation method. As discussed in Section 3.4, valid recommendation sets are obtained from the conjunction of the several recommendation sets, correspondent to each situation, taking into account the most restricted reachable state of the recommendation points. At the beginning, this operation results in the sequential recommendation t_1 (event New complaint received). Next, the recommendation set is reduced to the insertion of the task t_2 (Open new case). These correspond, respectively, to the insertion of the model fragments labeled with R1 and R2 in the model.

At the third step, it is inserted the model fragment R3. This case illustrates nested situations, i.e., situations involving a set of domain objects which take part in as a choice set in another situation. In the Handling Complaint process, t_3 and t_4 in situation 11 is the choice set of situation 12. In this case, as in any case having a nested situation in a branch of a gateway recommendation, this branch is set to an empty flow and the inclusion of objects involved in the nested situation is left to next recommendations. Thus, the result in the example is the inclusion of the model fragment R3 ($\times(t_9;)$). The insertion of the model fragment R9 is a consequence of the situation 10 and R10 is a gateway convergence caused by occurrence of the empty recommendation at both branches of the XOR gateway of R9.

Comparing the Event Processing Chain (EPC) [13] model obtained from the application of Mendling guidelines with the BPMN model showed in Figure 4, it can be observed that they mostly differ from the fact that the BPMN model was obtained considering active flow objects and situations which were explicitly described in the process textual description. Nevertheless, the resulting models are similar in terms of structure, so that the BPMN model, in this case, can be considered a good result accordingly to such guidelines. For example, the BPMN model is structured (Mendling's guideline 4) and minimizes routing paths (Mendling's guideline 2).

t1. New complaint received (evt)	**t7.** Archive Referral
t2. Open new case	**t8.** Inform Complainant
t3. Refer complaint Internally	**t9.** Complaint analysis
t4. Refer complaint externally	**t10** Archive Complaint
t5. Put Internal Referral on incident Agenda	**t11.** Contact Complainant
t6. Confirm External Referral	**t12.** Follow up Complainant

Table 2: Domain of the Handling Complaint Process

1. $t_2 \lhd t_1$	7. $t_8 \lhd (t3 \vee t_4)$
2. $t_3 \lhd t_2$	8. $t_8 \lhd t_7$
3. $t_4 \lhd t_2$	9. $(t_{10}, t_{11}) \lhd t_9$
4. $t_5 \lhd t_3$	10. $t_{12} \unlhd t_9$
5. $t_6 \lhd t_4$	11. $t_9 \otimes (t_3, t_4)$
6. $t_7 \lhd (t_3 \vee t_4 \vee t_9)$	12. $t_3 \otimes t_4$

Table 3: Situations for the Handling Complaint Process

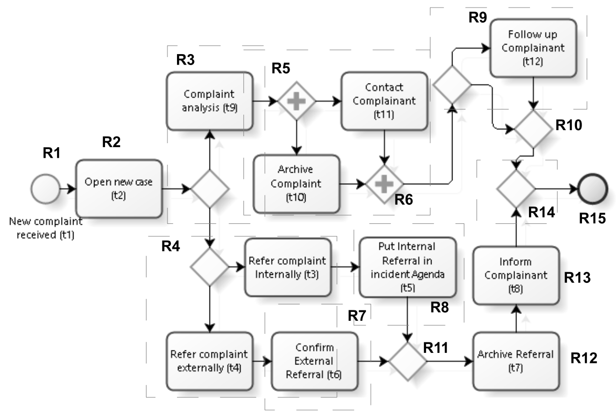

R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
New complaint received (t1)
Open new case (t2)
Complaint analysis (t9)
Archive Complaint (t10)
Contact Complainant (t11)
Follow up Complainant (t12)
Refer complaint Internally (t3)
Put Internal Referral in incident Agenda (t5)
Refer complaint externally (t4)
Confirm External Referral (t6)
Archive Referral (t7)
Inform Complainant (t8)

Figure 4: Handling Complaint Process BPMN Model.

5. LIMITATIONS AND EXTENSIBILITY ISSUES

The recommendation patterns discussed in Section 3 are able to support the modeling of linear business processes with the syntax limited by the pattern recommendations. Loops are not included on this previous catalog. Parallel

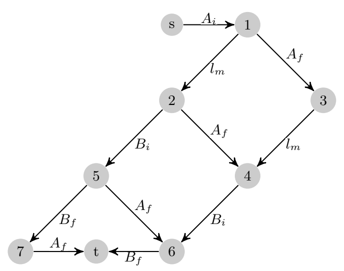


Figure 5: Hasse Diagram for the minimum Start-Start Time Lag Pattern.

situations are also not supported as discussed in Section 3.1. The application of the methodology was conducted to model example processes with different sizes and complexity including problems provided by the Business Process Management Initiative (BPMI) and problems from a real Human Resources Department (HRD) of a Brazilian educational institution. As it was expected, models showed compliance to the situation based conceptual view. However, such a compliance does not eliminate the need for validation. Actually, a lesson learned from such experience is that both process descriptions (situation based and imperative model) can be viewed as complementary, in the sense that resulting imperative models can also feed and help to improve the conceptual process view.

The methodology was manually conducted using a typical BPMN modeling tool and spreadsheets to register situations. However, it is expected a more feasible process with a computer based tool able to automatically generate recommendations.

Recommendation Patterns can be extended and set up accordingly to problem domain needs. An extension example can be showed by the introduction of a situation described as the minimum start-start Time Lag pattern discussed by Lanz *et al* [15]. This pattern describes the situation where a minimum time lag of specific duration should occur between the start time of two activities, let's say A and B. To model the behavior of this situation the involved activities were decomposed in terms of its initial and final events, and a *Lag* activity (l_m) was also introduced. The involved domain is given by D=$\{A_i,\ A_f,\ B_i,\ B_f, l_m\}$.

Figure 5 shows the Hasse diagram for the minimum start-start Time Lag situation behavior. Table 4 shows the recommendation sequences which can be inferred from this behavior. Each recommendation sequence represents an ordered sequence of recommendations which can be taken in order to fulfil the minimum time lag start-start situation.

1. $(A_i, l_m, B_i, B_f, A_f)$	4. $(A_i, +(A_f, l_m), B_i, B_f)$
2. $(A_i, A_f, l_m, B_i, B_f)$	5. $(A_i, l_m, +(B_i, A_f), B_f)$
3. $(A_i, l_m, A_f, B_i, B_f)$	6. $(A_i, l_m, B_i, +(B_f, A_f))$

Table 4: Recommendation Sequences for the Minimum Start-Start Time Lag Pattern.

Note that recommendations are given in terms of initial and final activity events. However, model structure options

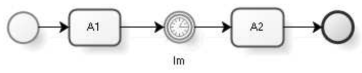


Figure 6: BPMN Model Equivalent to the Recommendation Sequence 2 of Table 4.

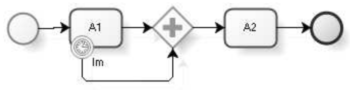


Figure 7: BPMN Model Equivalent to the Recommendation Sequence 4 of Table 4.

can be obtained by determining language specific constructions whose semantics is in conformity with one of these recommendations. For example, Figure 6 and 7 show two BPMN models correspondent to, respectively, the recommendation 2 and 4 of Table 4.

6. CONCLUSION

In this work the issue of Business Process Imperative Modeling support was approached by the so-called Recommendation Patterns and by a methodology based on these patterns. By using the proposed methodology, finding imperative modeling options (recommendations) is delegated to a systematic approach, so that modelers can concentrate in deciding among these options. Since recommendations can be defined accordingly to the modeling assumptions of choice, e.g., language specific aspects and modeling guidelines, it was also observed that following specific modeling rules and directives can also be supported by the methodology.

The application of the methodology showed model results with compliance to the conceptual Situation-based process description. However, it was observed that a model validation routine is still required and the Situation-based and the Imperative process descriptions should be viewed as complementary. Since the methodology still does not have computational support, determining recommendations produced a significant effort. Disregarding this effort, one can observe that the creation of BPMN models becomes easier. It is expected however, that, with the computational support, the methodology will be made easier and will require less effort. It is also expected, as future work, to investigate the cognitive aspects that permeate Process Modeling by observing difficulties such as how to decide among modeling options and how to trace the causal scope among modeling decisions and process quality parameters.

7. REFERENCES

[1] E. D. Adamides and N. Karacapilidis. A Knowledge Centred Framework for Collaborative Business Process Modelling. *Business Process Management Journal*, 12(5):557–575, 2006.

[2] A. Adi and O. Etzion. Amit- The Situation Manager. *The VLDB JournalŮThe International Journal on Very Large Data Bases*, 13(2):177–203, 2004.

[3] K. A. Baker, P. C. Fishburn, and F. S. Roberts. Partial Orders of Dimension 2. *Networks*, 2(1):11–28, 1972.

[4] I. Barba, C. Del Valle, B. Weber, and A. Jiménez. Automatic Generation of Optimized Business Process Models from Constraint-Based Specifications. *International Journal of Cooperative Information Systems*, 22(02), 2013.

[5] J. Becker, M. Rosemann, and C. Von Uthmann. Guidelines of Business Process Modeling. In *Business Process Management*, pages 30–49. Springer, 2000.

[6] I. T. Cameron and K. Hangos. *Process Modelling and Model Analysis*, volume 4. Academic Press, 2001.

[7] A. Caracaş and A. Bernauer. Compiling Business Process Models for Sensor Networks. In *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, pages 1–8. IEEE, 2011.

[8] N. N. Chan, W. Gaaloul, and S. Tata. Context-based Service Recommendation for Assisting Business Process Design. In *E-Commerce and Web Technologies*, pages 39–51. Springer, 2011.

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

[10] C. A. Ellis and G. J. Nutt. Office Information Systems and Computer Science. *ACM Comput. Surv.*, 12(1):27–60, Mar. 1980.

[11] S. Goedertier and J. Vanthienen. Declarative Process Modeling with Business vocabulary and Business Rules. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 603–612. Springer, 2007.

[12] S. Junginger and E. Kabel. Business Process Analysis. In *eBusiness in Healthcare*, pages 57–77. Springer, 2008.

[13] Y.-G. Kim. Process modeling for bpr: event-process chain approach. *ICIS 1995 Proceedings*, page 11, 1995.

[14] A. Koschmider, T. Hornung, and A. Oberweis. Recommendation-based Editor for Business Process Modeling. *Data & Knowledge Engineering*, 70(6):483–503, 2011.

[15] A. Lanz, M. Reichert, and B. Weber. Process Time Patterns: A Formal Foundation. *Information Systems*, 57:38–68, 2016.

[16] Y. Li, B. Cao, L. Xu, J. Yin, S. Deng, Y. Yin, and Z. Wu. An Efficient Recommendation Method for Improving Business Process Modeling. *Industrial Informatics, IEEE Transactions on*, 10(1):502–513, Feb 2014.

[17] J. Mendling, H. A. Reijers, and W. M. van der Aalst. Seven Process Modeling Guidelines (7pmg). *Information and Software Technology*, 52(2):127–136, 2010.

[18] OMG. Business Process Modeling Notation. Specification Version 2.0. Technical report, Object Management Group, March 2011.

[19] M. Pesic, H. Schonenberg, and W. M. van der Aalst. Declare: Full support for Loosely-Structured Processes. In *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*, pages 287–287. IEEE, 2007.

[20] A. Polyvyanyy, L. García-Bañuelos, and M. Dumas. Structuring Acyclic Process Models. *Information Systems*, 37(6):518–538, 2012.

[21] M. Rospocher, C. Ghidini, and L. Serafini. An Ontology for the Business Process Modelling Notation. In *FOIS*, pages 133–146, 2014.

[22] S. Sadiq, G. Governatori, and K. Namiri. Modeling Control Objectives for Business Process Compliance. In *Business process management*, pages 149–164. Springer, 2007.

[23] H. Schonenberg, B. Weber, B. van Dongen, and W. van der Aalst. Supporting Flexible Processes through Recommendations Based on History. In *Business Process Management*, pages 51–66. Springer, 2008.

[24] W. M. van Der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow Patterns. *Distributed and parallel databases*, 14(1):5–51, 2003.

[25] H. J. Wang and H. Wu. Supporting Process Design for e-Business via an Integrated Process Repository. *Information Technology and Management*, 12(2):97–109, 2011.