Reuse of Semantic Business Process Patterns

Lobna Makni[1], Nahla Zaaboub[1] and Hanene Ben-Abdallah[2]
[1]*Mir@cl Laboratory, Faculty of Economic Sciences and Management, Sfax University, Sfax, Tunisia*
[2]*Faculty of Computing and Information Technology, King Abdullaziz University, Jeddah, K.S.A.*

Keywords: Reuse, SB2P Repository, Search Method, Business Object, Instantiation Operator.

Abstract: Reusing business process artefacts, in business process modelling, can accelerate the design activity and improve the quality of the resulting products. Among the variety of reusable artefact types proposed during the last decade, *Semantic Business Process Pattern* (SB2P) offers business patterns that encapsulate knowledge and expertise in a particular domain. The reuse of a SB2P provides for the adoption of commonly used practices while offering guidance on possible variations. This paper proposes a reuse method that assists a business process designer, first, in retrieving an appropriate SB2P from a repository, and then adapting it through a set of instantiation operators to meet its specific requirements. The retrieval method relies on both semantic and behavioural information encapsulated in the SB2P. In addition, the instantiation operators ensure the derivation of a business process design that respects a set of quality metrics. The reuse method is illustrated through an example.

1 INTRODUCTION

The advent of reuse oriented modelling in BPM led to the proposition of several reusable artefact types. The various reusable artefacts (Haddar et al., 2012) rely on proven practices and solutions to provide for assistance in designing high quality business process models while reducing the designer costs.

The benefits of reuse in PAISs (Process Aware Information Systems) are widely accepted in theory and practice (Fettke and Loos, 2003). In fact, designing high quality process models from scratch is often time consuming, error-prone and costly (Indulska et al., 2009). In the last decade, a variety of reusable artefacts are proposed in the literature. Recently, (Makni et al., 2011) propose a new pattern concept called Semantic Business Process Pattern (SB2P). A SB2P is a pattern synthesized from a set of process models belonging to the same business domain. It is composed of process fragments that are semantically common among this set of process models. A SB2P is intended to accelerate the design process by providing the designer with a business process model skeleton containing all mandatory activities and some configurable ones (Haddar et al., 2012). It is *automatically* generated from a reduced set of business process models belonging to the same domain. SB2Ps are stored in a pattern repository organized according to business domains. The size of the SB2P repository can be important since many SB2Ps may be generated for the same domain.

Given a repository of SB2Ps, a designer would need a means, first, to retrieve the SB2P that covers his/her business requirements and, secondly, to instantiate a retrieved pattern to model his/her business needs in order to accelerate the design process and ameliorate the quality of the resulting process models. Existing business process retrieval methods use query languages (Awad, 2007), (Choi et al., 2007), (Jin et al., 2010) based on the control-flow perspective; the remaining aspects of a business process like data is, however, ignored. In addition, querying requires a good knowledge of the querying language. Furthermore, the so-far proposed instantiation operators (La Rosa et al., 2013), like add/remove a node, are generic operators intended for model evolution, and they do not explicitly analyse their effects on the model quality.

In this work, we contribute to business process modelling by assisting in the reuse of SB2Ps. More specifically, we propose a retrieval method for querying SB2Ps based on the data of business processes. In addition, we propose a set of instantiation operators applicable under constraints that accounts for the quality of the derived business process model.
The rest of the paper is organized as follows: Section

Makni L., Zaaboub N. and Ben-Abdallah H..
Reuse of Semantic Business Process Patterns.
DOI: 10.5220/0005003500360047
In *Proceedings of the 9th International Conference on Software Engineering and Applications* (ICSOFT-EA-2014), pages 36-47
ISBN: 978-989-758-036-9

2 positions our research in the light of related works. Section 3 reviews the SB2P concept through an example. Section 4 presents a SB2P retrieval method based on the data aspect. Section 5 presents a set of instantiation operators to adapt a SB2P to meet a designer's requirements. Section 6 summarizes the presented work and highlights its extensions.

2 RELATED WORK

Several solutions have been investigated to support the reuse of business process models. We classify these solutions into two categories: (i) reusable business process artefact retrieval and (ii) adaptation of reusable artefacts.

2.1 Reusable Business Process Artefact Retrieval

Mindful of the importance of query languages for business process models, the Business Process Management Initiative (BPMI) proposed to define a standard business process model query language in 2004. However, no standard has been published so far. Several approaches were proposed to express and execute queries over a collection of business process models (Beeri et al., 2008), (Awad and Sakr, 2012), (Jin et al., 2011), (ter Hofstede et al., 2013). They are classified into two main categories. The first category enables the formulation of a query as a business process model fragment (Jin et al., 2010), (Choi et al., 2007), (Jin et al., 2011), (ter Hofstede et al., 2013). Its aim is to identify all models in a repository, that contain the query fragment by detecting similar activity labels using *only* the equivalence semantic relation.

On the other hand, the second category provides specific constructs for expressing process model queries like existence or absence of paths between activities (Beeri et al., 2008), (Awad and Sakr, 2012). We cite, for example, the graphical query language called BPMN-Q (Business Process Modelling Notation-Query) proposed by (Awad and Sakr, 2012). It is a visual and graphical query language which can be used for querying a repository of process models modelled with BPMN standard. It extends BPMN notation with seven additional constructs *e.g.,* generic join to present the query purpose. It allows representing a structural BPMN queries and specifies whether a given process model is syntactically similar to a query (Awad, 2007), (Awad et al., 2008). In addition, queries can be formulated using control-flow aspect of process model in (Awad and Sakr, 2012), (Choi et al., 2007) (Jin et al., 2010), (Jin et al., 2011). Other aspects are treated, only in (Choi et al., 2007), like resources, process goal.

The execution of a query on a collection of process models can be handled in one of two ways: 1) by translating the query to SQL (Awad, 2007) or XML standard (Choi et al., 2007), (Beeri et al., 2008) or 2) by using the sub-graph isomorphism algorithms (Jin et al., 2010). In this second case, indexes can also be used to speed up the query execution (Jin et al., 2013).

Note that there are other techniques available, in the literature, which can be useful for querying process model repositories like VisTrails and WISE systems. The VisTrails system (Scheidegger et al., 2008) allows users to query workflow by example and to refine workflows by analogies. The workflow engine WISE (Shao et al., 2009) returns the most specific workflow hierarchies containing matching keywords.

Overall, the proposed query languages proposed in the literature focus only on control-flow aspect of business process and ignores other process aspect like data. In this paper, we propose a new method to retrieve a SB2P like BPMN process model from a repository of patterns using the semantic (business object), behavioural (causality and concurrency relations) and hybrid (combination of semantic and behavioural) criteria on the basis of textual query.

2.2 Adaptation of Reusable Artefacts

When creating business process models, one of the fundamental challenges the process designer faces is to cope with the adaptation of reusable artefacts which are complex and of coarse granularity such as reference models. Indeed, a reference model has a recommending character, covers a family of process models, and can be customized in different ways according to the designer requirements. Several approaches have been proposed in the literature to adapt reference models through configuration (Rosa, 2009), (Rosemann and van der Aalst, 2007), (Gottschalk et al., 2008). They are classified into two main categories. The first category is based on behaviour by extending the existing process modelling language. In this context, configurable process models have been developed to make reuse of reference process models with managing business process variability (Rosa et al., 2009), (Rosa, 2009). For example, Configurable-Event Process Chain (C-EPC) provides a support to customize reference models where EPC functions can be annotated. The annotation provides the information that these functions are mandatory or optional (Rosemann and van der Aalst, 2007), (Rosa et al., 2007).

In addition, (Gottschalk et al., 2008) propose an approach for configuring workflow models by enabling, hiding or blocking some configurable workflow elements. These approaches allow to define requirements over the configuration alternative that may be chosen. They neither are allowed to move or add model elements nor to adapt element attributes when configuring a variant. So, requirements do not prescribe mandatory constraints, but serve as recommendations (i.e., two activities either may have to be deleted together from the reference process or none of them). The second category of approaches is based on structure through adaptation patterns. The adaptation depends on concrete designer requirements by applying a set of change operations. Weber et al. (Weber et al., 2008) provide a comprehensive overview of possible adaptation patterns that can occur when process models are modified. An adaptation pattern represent generic operations which can be applied on any process model. It enables structural changes of process schemes. For example, AP1 (AP2) allows inserting (deleting) a process fragment in a process model. The Provop proposal follows this approach (Hallerbach et al., 2010). It provides advanced tool support for adapting a base process and for ensuring syntactical and semantical correctness of derived process variants (Hallerbach et al., 2010).

3 OVERVIEW OF THE SB2P CONCEPT

A SB2P is a pattern synthesized from a set of business process models belonging to the same business domain (Makni et al., 2010). Informally, it is composed of process fragments semantically common among the source models.

These fragments may have different structural and/or behavioural representations in the original models; the SB2P factorizes these constructs into a form that preserves the semantics of these fragments. It is intended to accelerate the design process by providing the designer with a business process model skeleton containing all mandatory activities and some configurable ones. Mandatory activities are recurrent and equivalent activities shared between the source models (e.g., ”revaluation at actual price”, ”WIP calculation”). Together with their associated control-flow, they constitute the backbone of the pattern and have to be present in any business process model derived from this SB2P. For this reason, they are coloured pink to highlight their importance to the designer in the sense when removing one of these activities from the pattern it may change its semantics and corrupt the process behaviour. Also, they play the role of milestones, between which there are variation points. We illustrate the SB2P concept by the example of figure 1-a which shows a pattern automatically constructed from two business process models choosen among the SAP reference models (Keller and Teufel, 1998). These models are from the domain of cost accounting which computes the costs incurred in the production of company activities (e.g., internally manufactured materials) in order to make complex cost analysis.

In a SB2P, there are also variation points. A variation point is a precise position within a SB2P that admits different structural representations in the source models. For example, we can found a unit of work represented in one process model by a single activity (e.g., ”manufacturing order overhead calculation”) but the same unit of work is represented by a *process fragment* composed of a collection of activities (e.g., ”cost object overhead calculation” and ”product cost collector overhead calculation”) in the other process model. These structural differences are extracted from the source models during the SB2P construction process. They are stored in a database and are linked to their corresponding variation point. So, a variation point have multiple variants and a decision needs to be taken on which variant to use when instantiating the SB2P. For example, in figure 1-a the variation point C can be instantiated with one of the two fragments PF1 and PF2 of figure 1-b. PF1 and PF2 were extracted from the source models during the SB2P construction process. Once all the variation points have been treated, the SB2P becomes a derived business process model.

In a SB2P, a variation point can be a:

- Refined Activity: It stands for a unit of work which may be represented by a process fragment which subsumes and refines it. In the pattern, the refined activity is represented as an annotated and empty sub-process. The annotation contains a list of refining fragments. When instantiating the pattern, the designer can either keep the activity as a single unit, or refine it into one of its annotating process fragments.
- Decomposable Activity: It stands for a unit of work which may be represented by a process fragment which partly-corresponds and decomposes the decomposable activity. In a SB2P, the decomposable activity is represented as an annotated and empty sub-process. The annotation contains a list of component fragments. When instantiating the pattern, the designer can either keep the activity as a single unit, or decompose it into one of its annotating process fragments.

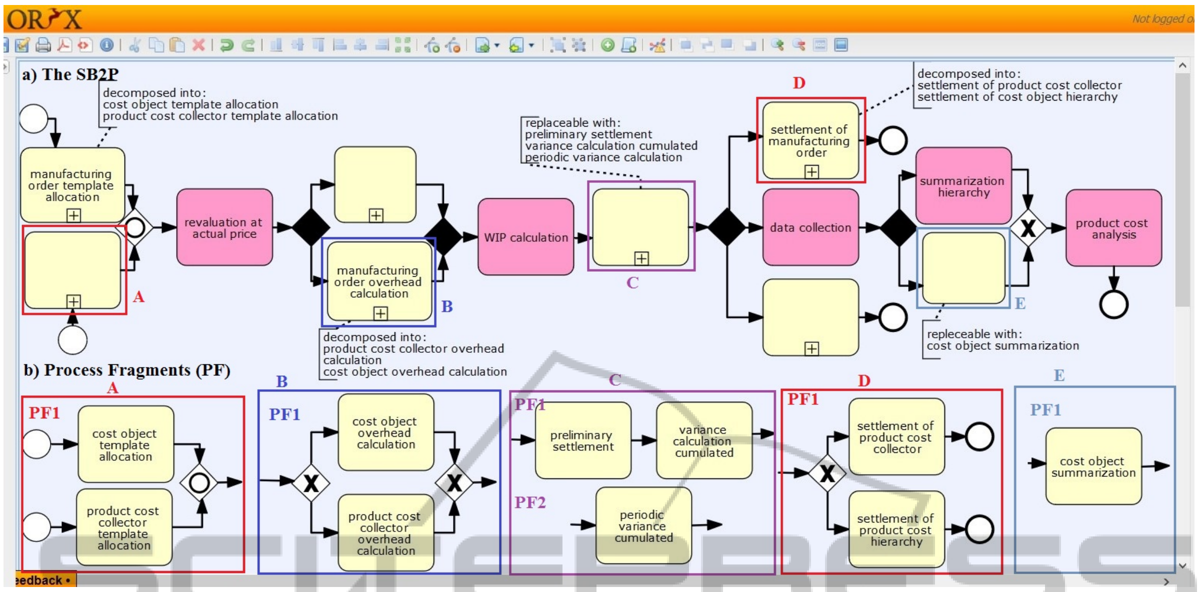

ORYX
Not logged o
a) The SB2P
decomposed into:
cost object template allocation
product cost collector template allocation
manufacturing order template allocation
revaluation at actual price
A
manufacturing order overhead calculation
B
decomposed into:
product cost collector overhead calculation
cost object overhead calculation
WIP calculation
replaceable with:
preliminary settlement
variance calculation cumulated
periodic variance calculation
C
D
settlement of manufacturing order
decomposed into:
settlement of product cost collector
settlement of cost object hierarchy
data collection
summarization hierarchy
product cost analysis
E
repleceable with:
cost object summarization
b) Process Fragments (PF)
A
PF1
cost object template allocation
product cost collector template allocation
B
PF1
cost object overhead calculation
product cost collector overhead calculation
C
PF1
preliminary settlement
variance calculation cumulated
PF2
periodic variance cumulated
D
PF1
settlement of product cost collector
settlement of product cost hierarchy
E
PF1
cost object summarization
eedback

Figure 1: An example of SB2P for the cost accounting process.

- Skipped Activity: It is a silent activity without added value. In the pattern, it is an optional activity and is represented as an annotated empty activity. When instantiating the pattern, the designer can insert into or delete a skipped activity, or insert in it one of its annotating process fragments.

- Configurable Connector: It is a generic connector. It subsumes possible build time connectors that are less or equally expressive. It is annotated (e.g. configurable OR, AND or XOR) and coloured black to differentiate it from other connectors in the pattern. It can have several variants (AND, XOR, OR) depending on its type.

As a SB2P resembles a business process model, its formal definition is also close to that of business process models. However , to consider variation points, we add a fourth one, the configurable connector. Also, we define the function F_{ref} which assigns process fragments to refinable activities. F_{decomp} is a function that assigns process fragments to decomposable activities. F_{skip} is a function that assigns process fragments to skipped activities.

To give a formal definition of a SB2P, we start with defining a process fragment.

Definition 1 (Process-fragment): A process fragment is a tuple (O, A, G, F) where:

- O is a set of objects partitioned into a disjoint sets A of non-empty activities (task or sub-process) and gateways G.

- $F \subseteq O \times O$ is the control-flow relation between objects.

Definition 2 (SB2P): A SB2P is a tuple $(O, A, E, G, F, F_{ref}, F_{decomp}, F_{skip})$ where:

- $O = A \cup E \cup G$ is a set of objects partitioned into disjoint sets of activities A, events E, and gateways G; $A = A^M \cup A^S \cup S$ is the set of activities partitioned into a non-empty set of mandatory activities A^M, A^S a set of skipped activities and a set of sub-processes S; $G = G_F \cup G_J \cup G_X \cup G_M \cup G_C$ is a set of gateways partitioned into disjoint sets of parallel fork gateways G_F, parallel join gateways G_J, XOR decision gateways G_X, XOR merge gateways G_M and configurable gateways G_C.

- $F \subseteq O \times O$ is the control-flow relation used to define process fragments, *i.e.,* a set of sequence flows connecting objects.

- $F_{ref}: S \to \mathcal{P}(T \cup Fg)$ where S is the set of sub-processes and $\mathcal{P}(T \cup Fg)$ denotes the power-set of tasks T and process fragments F. For $sp \in S$ and $t \in \mathcal{P}(T \cup Fg)$, $F_{ref}(sp) = t$ means that each element of t refines sp.

- $F_{decomp}: S \to \mathcal{P}(T \cup Fg)$. For $sp \in S$ and $t \in \mathcal{P}(T \cup Fg)$, $F_{decomp}(sp) = t$ means that each element of t is a component of sp.

- $F_{skip}: A^S \to \mathcal{P}(T \cup Fg)$. For each $sk \in A^S$ and $t \in \mathcal{P}(T \cup Fg)$, $F_{skip}(sk) = t$ means that each element of t can replace sk.

4 SB2P RETRIEVAL METHOD

As mentioned in Section 2, the various query languages to search for reusable business process artefacts (Awad and Sakr, 2012), (Beeri et al., 2008), (Jin et al., 2011), (ter Hofstede et al., 2013) do not account for the data aspect. To overcome this shortage, we propose a search method that exploits the business objects which model the data aspect. Before introducing our method, we first define the SB2P repository structure.

4.1 SB2P Repository

Figure 2 shows the conceptual model of a SB2P repository which is an extension of the UML meta-model of BPMN (OMG, 2013). Note that SB2Ps (class Pattern) and BPMN business processes (class Process) are quite similar because both are composed of activities, events and gateways which are related by sequence flows. So, the extending elements concern the domain and the variation points. In fact, SB2Ps are organized by domain. Each SB2P is composed of a set of nodes. A node can be an event, a gateway or an activity. A gateway may be ”OR”, ”AND” or ”XOR”. An activity is characterized by a label and acts on an data object. We also characterise an activity by its type. Indeed, an activity, in a SB2P, is either mandatory or configurable. A configurable activity is a variation point in a SB2P on which the designer can introduce some modifications (i.e. by applying instantiation operators (see section 5) to imitate the pattern according to his/her requirements. It can be refined, extended (have additional dependency) or decomposed. An activity, in a SB2P, is related to one or more activities. The type of behavioural relationship is either causal (means that an activity follows the other) or parallel.

To improve the efficiency of SB2Ps retrieval within the repository, we define two inverted *indexes*, namely, the *activity index* ActIndex and the *business object index* BOIndex. The business object index implements the association between the classes Pattern_Acitivity and DataObject in the UML meta-model of figure 2. The business object index takes the form of set of pairs *(business object; activity list)* where *business object* denotes the object appearing in some activity labels and *activity list* denotes the set of these activities. When a new pattern is added to the SB2P repository, all the activity labels can be extracted, and then the business object index can be updated.

On the other hand, the activity index implements the association between the classes Pattern_Acitivity and Activity in the UML meta-model of figure 2. The activity index takes the form of set of pairs *(activity; pattern list)* where *activity* denotes an activity occurring in some patterns and *pattern list* denotes the set of these patterns. Given a pattern, we first extract all its activities. The extraction algorithm takes linear time in terms of the number of activities of the pattern. Then, the mapping between the activities and the patterns is updated in the activity index.

4.2 Search Methods

Our search method is based on three criteria which are the semantic, the behavioural and the hybrid (combination of the two cited criteria) one.

4.2.1 Semantic Search

The SB2P semantics is coded in its activities' labels. The label of an activity defines the role of the activity in the process pattern and the type of business objects on which they act. Thus, we propose to exploit activity labels and especially business objects to make semantic search. Indeed, a business object refers to a concept from the enterprise business domain. It describes an entity manipulated by the activities and therefore by the business actors. Therefore, a search method based on business objects allows an accurate query of the SB2Ps repository according to semantic criteria.

Our method extracts *patterns* which contain activities acting on business objects whose names are introduced by the designer in the query. A query is a predicate composed of business objects and semicolons which replace the logical operator 'and'.

Semantic criteria based on business objects can improve research results. In fact, they allow the detection of all activities acting on the business object without the need to make neither calculi of similarity between activity labels nor thresholds to compare results. Moreover, we can detect not only equivalent business objects but also other semantic relationships like hyponymy and hypernymy in order to enlarge our research results. In addition, using business objects as search criteria accelerates the search process because the number of business objects is smaller than the number of activities in the organization. In addition, the use of indexes on business objects accelerates and make the search ore efficient.

To compare business objects, we need a business domain specific ontology. In our study, we construct a Business Object ONtology (BOON) which offers a reference terminology that defines semantic correspondences between business objects. The BOON has a WordNet like conceptual schema an a logical

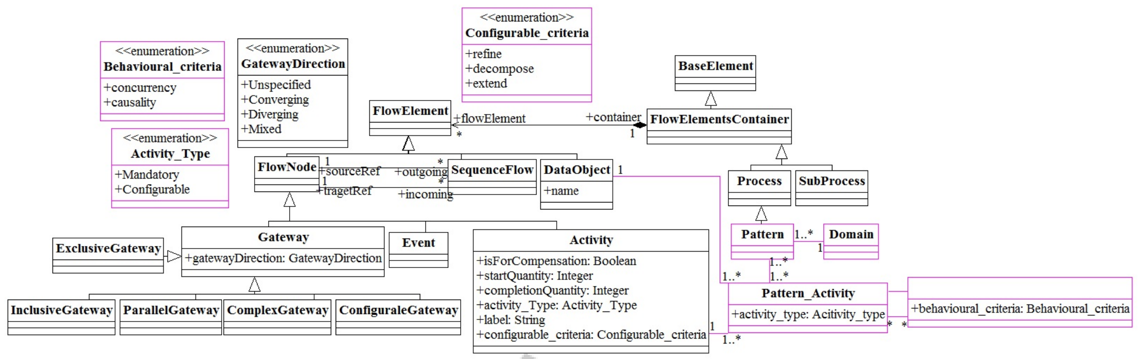


Figure 2: Conceptual model of SB2P repository.

reasoning. But, it is domain specific and it takes into consideration synonymy, hyponymy, hypernymy, meronymy and holonymy relations. Further, unlike WordNet, it accepts composed nouns like ”manufacturing order”. For example, using the BOON, we can detect that in the manufacturing domain the business object ”manufacturing order” is equivalent to the business object ”Order”.

The query is answered via a set of resolution phases. The first phase consists in scanning the business object index to check whether the in-putted BO name is associated with any SB2P. If the search leads to one or more SB2P(s) in the result, we show the detected patterns to the designer. Otherwise, we carry on with the search within the business objects derived from the in-putted one using synonymy, subsumption and part-of semantic relationships. If all phases fail, the query processing is terminated with no match.

```
Procedure SemanticSearch (Pattern[] SB2Ps,
         DataObject BO, Pattern[] ResultP)
i, j, k: integer
SynBO, SubBO, PartBO: DataObject[]
Begin
if (BO in BOIndex) then
  ResultP = list of patterns associated to BO
  else
   SynBO <- getSynonym(BO)
   For i=1 to |SynBO|
   {
    if (SynBO(i) in BOIndex) then
     ResultP=list of patterns associated to BO
   }
   SubBO <- getSubsumption(BO)
   For j=1 to |SubBO|
   {
    if (SubBO(j) in BOIndex) then
     ResultP=list of patterns associated to BO
   }
   PartBO <- getMeronym(BO)
   For k=1 to |PartBO|
   {
    if (PartBO(k) in BOIndex) then
     ResultP=list of patterns associated to BO
   }
  EndIf
End.
```

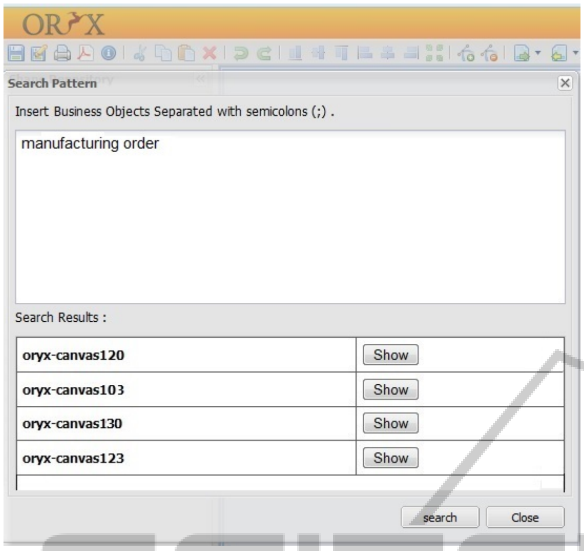
We have developed a tool, called ’SB2P Reuse Tool’, which automates our SB2P search method. ’SB2P Reuse Tool’ is a plug-in integrated in the Oryx platform (Oryx-Editor, 2010). To search for a pattern, the user starts with entering the domain of the pattern. Then, he gives a key word like query by specifying a list of business objects separated by semicolons. Figure 3 represents the result of the execution of the query ”manufacturing order” in the manufacturing domain. In this figure, we see that there are four patterns (oryx-canvas 120, oryx-canvas 103, etc.) in the repository which use the ”manufacturing order” business object. The designer clicks on the button ’Show’ in front of each pattern to show it.

Semantic search using business objects is efficient and not complex, because the number of business objects of a given domain is not high. Moreover, it is of great help for the designer since he/she can put a synonym, a subsuming or a component business object in the query. Using business objects as search criterion can be also adopted for searching for specific models within repositories of business process models.

However, a query restricted to expressing business objects is not always sufficient. In fact, one may need to express behavioural relationships between activities in a query like causality or concurrency to define the context within which some activities should be performed in the desired pattern. For this reason, we add in the next section behavioural criteria to our search method.

4.2.2 Behavioural Search

The SB2P notation presents a behavioural description of business processes through control-flow representation. A SB2P can be regarded as a graph. So, it represents the order in which activities are performed during the execution of the process. When we search


Search Pattern

Insert Business Objects Separated with semicolons (;) .

manufacturing order

Search Results :

oryx-canvas120	Show
oryx-canvas103	Show
oryx-canvas130	Show
oryx-canvas123	Show

search Close

Figure 3: Semantic search.

for a SB2P in the repository, we sometimes want to express a behavioural relationship between activities like causality or concurrency. In this paper, we concentrate on these two relationships because these are the most important behavioural operators. So, we start with defining them:

- Causality: Two activities A and B in a SB2P are in causal relation, if the graph of the SB2P contains a path with at least one arc leading from A to B (A precedes B).
- Concurrency: Two activities A and B are in concurrency relation, if they are not causally related and they have a common ancestor which must be a parallel connector (A is in parallel with B).

These basic behavioural operators are sufficient to express behavioural search criteria. In fact, the SB2P subsumes the behaviour of a great number of models. So, we don't need to express sharp behavioural queries. To answer a behavioural query, we define the function *behavSearch()* which takes as input two activity labels and a behavioural criterion (0 for *Causality* and 1 for *Concurrency*). It returns an array of all patterns of the repository satisfying the in-putted query.

On the other hand, we adopt the approach proposed by (Jin et al., 2011) to detect behavioural relations between activities. This approach is based on two phases: extraction of behavioural relations and building indexes on behavioural relationships. We can check whether a task can be executed (parallel or causal) by traversing all the transitions in the pattern. We develop the following algorithm *ExtC* to detect causal relations between pairs of activities in the SB2P. This algorithm consists in a procedure that takes as input the SB2P pattern and its activities. The purpose of this procedure is to extract behavioural causal relationships between each pair of activities in a SB2P.

```
Procedure ExtC (Pattern p, Pattern_Activity[]a)
 i:  integer
 NodeSucc: Pattern_Activity
Begin
 For{i = 1 to |a|}{
   NodeSucc <- getSucc(a[i])
   Causality(p,a[i] ,NodeSucc)
 }
End.
```

Then, we can update the inverted index *ActIndex*. Every entry of the index is composed of a pair of activities, a character that indicates the type of behavioural relation between them and the list of pattern where the activities appear and satisfy the indicated relationship. We choose the character 'S' for causality and 'C' for concurrency. The following algorithm of the procedure Causality stores the causality ('S') behavioural relationship.

```
Procedure Causality(Pattern p,Pattern_Activity
                    node, Pattern_Activity
                    succNode)
 i, j:  integer
 NodeSucc: Pattern_Activity[]
 Begin
 If(node ofType Activity)
   {
     If(succNode ofType Activity)
      {
         updateActIndex((node, succNode,'S'),p)
      }
     else
      {
         NodeSucc <- getSucc(succNode)
         For(j = 1 to |NodeSucc|){
           Causality(p,node,NodeSucc[j])
         }
      }
   }
 End.
```

The complexity of the algorithm *ExtC* is calculated in the worst case on the basis of the loop and the nested recursive call of the procedure *causality*(). Thus, the complexity of the algorithm is $O(n^2)$.

To detect concurrency relationships between activities in the SB2P, we define the procedure *ExtConcurrency*. This latter detects all pairs of activities in the pattern which have a parallel connector (AND) as a common ancestor.

```
Procedure ExtConcurrency (Pattern  p,
                          Pattern_Activity[] a)
i, j: integer
Begin
For(i = 1 to |a|-1)
```

```
 {
   For(j = i+1 to |a|)
     {
       If(a[i]!=a[j])
          {
            If(commonAncestor(a[i], a[j],p)
            {
              updateActIndex((a[i], a[j],'C'),p)
            }
          }
     }
 }
End.
```

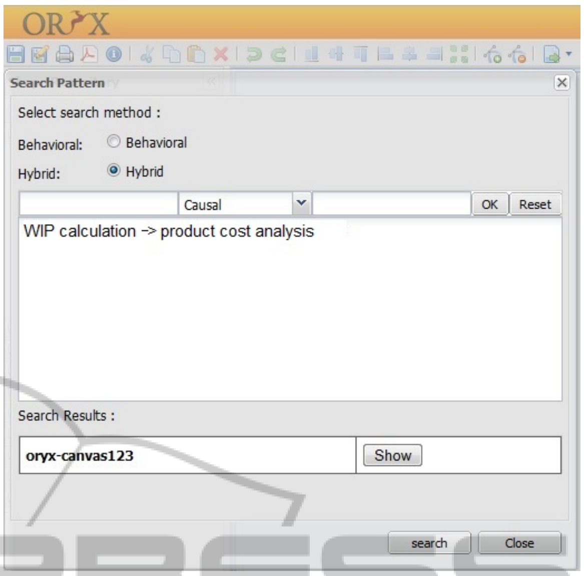
When a new pattern is added to the repository, all behavioural relations are extracted and inserted into the index. It is easy to see that the index can be updated incrementally. When a pattern is deleted from the repository, we delete it from all inverted index entries.

4.2.3 Hybrid Search

The hybrid search combines the semantic and behavioural search methods. It is conducted according to the following algorithm:

```
Function Hybrid (String label1, String label2,
                 integer behavRel):Pattern[]
p: Pattern[]
DLabel1, DLabel2: String[]
Begin
  p <- behavSearch (label1,
  label2, behavRel)
  If ( p is not null)
  Then return p
  Else {
   DLabel1 <- getSyn(label1)+
   getSub(label1)+getPart(label1)
   D2Label2 <- getSyn(label2)+
   getSub(label2)+getPart(label2)
   For i = 1 to |DLabel1| {
     For j = 1 to |DLabel2|{
       p <- behavSearch(DLabel1[i],
         DLabel2[j], behavRel)
       If ( p is not null)
       Then return p
      }
    }
  }
  return p
End.
```

The idea behind this algorithm is to enlarge our search by deriving behavioural queries from the introduced one. The function *Hybrid* takes as input a pair of activity labels *label1* and *label2* and the behavioural relation behaveRel between them and returns the list of patterns containing these activities related by the in-putted behavioural relationship or other derived relationships. First, it checks if the behavioural search *BehavSearch* leads to one or more



Select search method :
Behavioral: ○ Behavioral
Hybrid: ● Hybrid
Causal | OK | Reset
WIP calculation -> product cost analysis
Search Results :
oryx-canvas123 | Show
search | Close

Figure 4: Hybrid search.

results. If not, it calls the functions *getSyn()* to get synonyms, *getsub()* to derive hypernyms, hyponyms and *getPart()* to have holonyms and meronyms from label1 and label2. The derived labels are obtained by applying the Semantic rules proposed in (Makni et al., 2012). They are based on the comparison of actions and business objects of the activity labels.

Figure 4 shows the result of the query defined in the text box devoted to the query specification. This query is composed of causally (causal behavioural operator) related activities ("WIP calculation" and "product cost analysis"). As shown in this figure, we obtain one pattern (oryx-canvas 123) which contains these two causally related activities. This corresponds to the pattern of figure 1-a.

The complexity of the hybrid search algorithm is calculated in the worst case on the basis of the functions *getSyn()*, *getSub()* and *getPart()*. The complexity of each one is $O(n^2)$. Therefore, the complexity of the overall algorithm is $O(n^2)$.

4.3 Empirical Validation

We have conducted an experiment to evaluate our search algorithms and to highlight the importance of the semantic search. To do this, we have automatically constructed a test collection of SB2Ps from the SAP reference models (Keller and Teufel, 1998). Then, we measured the precision and recall of our search algorithms using only the behavioural and hybrid criteria (without the semantic one). The precision is the number of correctly retrieved patterns divided by the number of retrieved patterns. Recall is the number of correctly retrieved patterns by the total

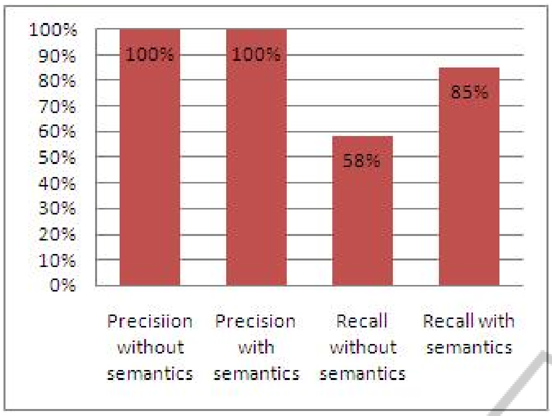


Figure 5: Precision and recall of our search algorithms.

number of patterns. Then, we compare the values of precisions and recalls.

The histogram of figure 5 show the results of our experiment. We see that all search algorithms give precise results. But the recall becomes high when we add semantics.

5 INSTANTIATION OPERATORS

An instantiation operator acts on a variation point of a SB2P to adapt it according to the designer's requirements. When all the variation points are treated by instantiation operators, a SB2P becomes a regular business process represented in BPMN. Depending on the variation point, a set of instantiation operators may be applicable. Each instantiation operator is subject to pre and post conditions that ensure that the final business process is syntactically well-formed and remains within the semantic scope of the original SB2P. In addition, none of the instantiation operators changes the behaviour of the SB2P's mandatory activities: being consensus activities among the process models of the domain, mandatory activities must be preserved. Furthermore, besides the syntactic well-formedness, semantic and behavioural preservation, the instantiation operators should guarantee the derivation of a process model with a quality not less than the SB2P's original one. In the remainder of this section, we present the five instantiation operators which are applicable to the various types of variation points in SB2Ps:

- *insert*: to add a process fragment in a variation point.
- *abstract*: to keep the variation point as it is without making any modification.
- *delete*: to remove an activity from the pattern.
- *rename*: to alter the name of an activity.
- *configure*: to specify the type of configurable connectors.

We will present the above operators classified according to the variation point types. For each instantiation operator, we present its pre and post conditions, its steps and results in the SB2P elements, and its impact on the mostly used business process model complexity metrics calculated by the quality evaluation tool proposed in (Makni et al., 2010). The considered metrics are the size, the density and the Control-Flow Complexity (CFC) metric.

The size metric counts the number of activities in a business process model (Cardoso et al., 2006). In addition, the CFC metric concerns the analysis of XOR-splits, OR-splits, and AND-splits control-flow elements. The main idea behind the metric is to evaluate the number of mental states that have to be considered when a designer is developing a process (Cardoso et al., 2006). The density metric relates the number of available connections to the number of maximum connections for the given number of nodes (Cardoso et al., 2006).

5.1 Instantiation of Decomposable or Refined Activities

A decomposable or refined activity can be subject to an *insert* or *abstract* instantiation operator.

5.1.1 Operator *insert*

- Pre-condition: The process fragment to be inserted is well-formed (*i.e.*, structured block (Nicola et al., 2010)).
- Post-condition: The SB2P is well-formed.
- Steps: Select one of the process fragments annotating the decomposable or refined activity, insert it into the variation point and eliminate the annotation from this activity.
- Results: The decomposable or refined activity becomes a regular sub-process object encapsulating the detailed activities selected by the designer.
- Effects on quality metrics: The size metric increases. Also, the density and the CFC metrics increase. Consequently, the overall architectural complexity of the resulting process model increases.

Figure 6 shows an example of instantiating the decomposable activity ”Manufacturing order overhead calculation” which corresponds to the variation point ’B’ in figure 1-a. The designer chooses the process fragment (PF1) from the set of process fragments ’B’

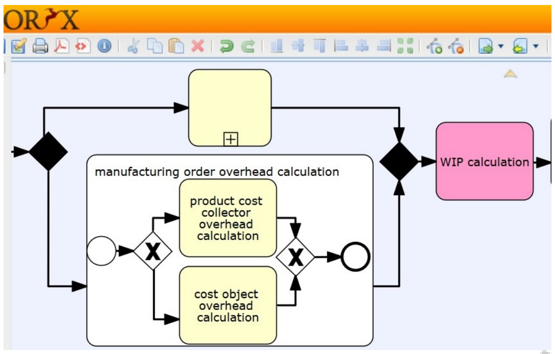

ORYX
manufacturing order overhead calculation
product cost collector overhead calculation
cost object overhead calculation
WIP calculation

Figure 6: Instantiation of decomposable activity.

(see figure 1-b) and applies the *'insert'* instantiation operator. As a result, the sub-process "Manufacturing order overhead calculation" contains the process fragment composed of "cost object overhead calculation" and "product cost collector overhead calculation".

5.1.2 Operator *abstract*

- Pre-condition: None
- Post-condition: None
- Steps: The annotation of the decomposable activity is eliminated.
- Results: Neither semantic nor structural modifications are introduced in the pattern.
- Effects on quality metrics: No changes.

5.2 Instantiation of Skipped Activities

A skipped activity can be subject to a Delete, Rename or Insert instantiation operator.

5.2.1 Operator *delete*

- Pre-condition: None.
- Post-condition: The SB2P is well-formed.
- Steps: Redirect each of the incoming edges of the skipped activity to all the targets of its outgoing edges; delete all of its outgoing edges; finally delete the skipped activity.
- Results: No semantic modification is introduced on the model.
- Effects on quality metrics: The size metric of the instantiated pattern decreases and also its density declines. Consequently, the overall structural complexity is reduced, hence, the instantiated pattern will be easier to understand.

5.2.2 Operator *rename*

- Pre-condition: The name to be inserted respects the syntactic pattern: action + business object (Makni et al., 2012).
- Post-condition: None.
- Steps: Select one activity from the list annotating the skipped activity to rename it or enter a new label. Then, eliminate the annotation from this activity.
- Results: No structural modification is introduced into the model.
- Effects on quality metrics: None of the structural metrics is affected. However, names which reveal the intention of the designers more clearly improve understandability of the model and consequently result in decreased costs of change and reduced errors (Becker et al., 2000).

5.2.3 Operator *insert*

- Pre-condition: The process fragment to be inserted is well-formed (*i.e.*, structured block) (Nicola et al., 2010).
- Post-condition: The SB2P is well-formed.
- Steps: Select one of the process fragments annotating the skipped activity, insert it into the skipped activity and eliminate its annotation, then rename the skipped activity.
- Results: The skipped activity becomes a regular sub-process containing the inserted process fragment.
- Effects on quality metrics: The size metric increases. Also, the density and the CFC metrics increase. Consequently, the overall architectural complexity of the resulting process model increases.

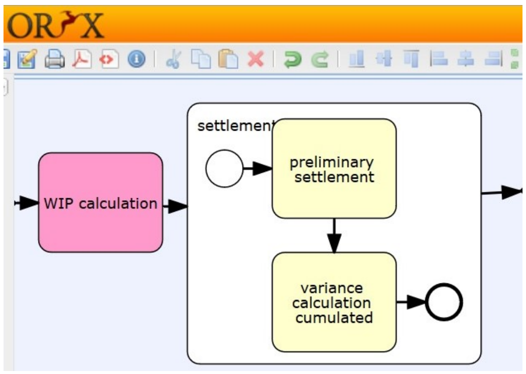


Figure 7: Instantiation of skipped activity.

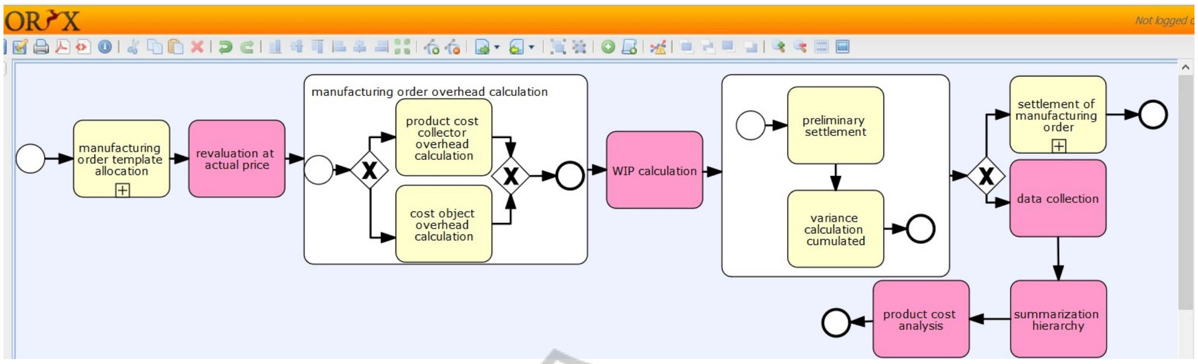

ORYX
Not logged
manufacturing order template allocation
revaluation at actual price
manufacturing order overhead calculation
product cost collector overhead calculation
cost object overhead calculation
WIP calculation
preliminary settlement
variance calculation cumulated
settlement of manufacturing order
data collection
summarization hierarchy
product cost analysis

Figure 8: An example of instantiated SB2P for the cost accounting process.

Figure 7 shows the instantiation of the variation point "C" (see figure 1-a) which is a skipped activity. in by applying the insert instantiation operator. As a result, the empty sub-process contains the process fragment (PF1) in the figure 1-b composed of "preliminary settlement" and "variance calculation cumulated".

5.3 Instantiation of Configurable Connectors

5.3.1 Operator Configure

- Pre-condition: None.
- Post-condition: None.
- Steps: A configurable OR-connector may be mapped to a regular OR-, XOR- or AND-connector. The configurable AND-connector may *only* be mapped to a regular AND-connector with a decision to be made by the designer as to how many of n available process paths are to be executed in synchronization. The configurable XOR-connector may *only* be mapped to a regular XOR-connector or to a single process sequence.
- Results: Each configurable connector will becomes a regular one.
- Effects on quality metrics: Evidently, gateways affect the Control Flow Complexity (CFC) of the pattern. The CFC is calculated by adding the CFC of all split constructs presented in the pattern. The greater the value of the CFC, the greater the overall architectural complexity of the pattern is.

Figure 8 is an example of an instantiated SB2P. We obtained it after applying the *'insert'* instantiation operator on the variation point 'B' which is decomposable activity (see figure 1-a) and on the variation point 'C' which is a skipped activity (see figure 1-a). In addition, we applied the *'delete'* instantiation operator on the skipped activities marked by 'C' and 'E' in figure 1-a and the *'abstract'* instantiation operator on the variation point decomposable activity marked by 'A'. Finally, we applied the *'configure'* instantiation operator on the configurable connectors. In summary, we used 6 instantiation operators ('insert', 'delete', 'abstract' and 'configure') to adapt the pattern.

6 CONCLUSION

In this paper, we presented a reuse method for SB2P, which assists a designer in retrieving a SB2P from a repository and applying a set of instantiation operators. The novelty of the proposed retrieval method is that it is based on the business objects while at the same time accounting for the behavioural aspect of a business process. To our knowledge, this work is the first in proposing this method. Besides the retrieval method, our reuse method defines a set of instantiation operators which are aware of the model quality. In future work, we aim to apply our SB2P construction method, instantiation operators and search algorithms on large and complex process models in order to study the impact of scalability.

REFERENCES

Awad, A. (2007). Bpmn-q: A language to query business processes. In *EMISA*, pages 115–128.

Awad, A., Polyvyanyy, A., and Weske, M. (2008). Semantic querying of business process models. In *Proceedings of the 2008 12th International IEEE Enterprise Distributed Object Computing Conference*, EDOC '08, pages 85–94, Washington, DC, USA. IEEE Computer Society.

Awad, A. and Sakr, S. (2012). On efficient processing of bpmn-q queries. *Computers in Industry*, 63(9):867–881.

Becker, J., Rosemann, M., and von Uthmann, C. (2000). Guidelines of business process modeling. In *In,: Business Process Manegement: Models, techniques and empirical studies.eds*, pages 30–49. Springer-Verlag.

Beeri, C., Eyal, A., Kamenkovich, S., and Milo, T. (2008). Querying business processes with bp-ql. *Inf. Syst.*, 33(6):477–507.

Cardoso, J., Mendling, J., Neumann, G., and Reijers, H. A. (2006). A discourse on complexity of process models. In *Proceedings of the 2006 International Conference on Business Process Management Workshops*, BPM'06, pages 117–128. Springer-Verlag.

Choi, I., Kim, K., and Jang, M. (2007). An xml-based process repository and process query language for integrated process management. *Knowledge and Process Management*, 14(4):303–316.

Fettke, P. and Loos, P. (2003). Classification of reference models: a methodology and its application. *Inf. Syst. E-Business Management*, 1(1):35–53.

Gottschalk, F., van der Aalst, W. M. P., Jansen-Vullers, M. H., and Rosa, M. L. (2008). Configurable workflow models. *Int. J. Cooperative Inf. Syst.*, 17(2):177–221.

Haddar, N. Z., Makni, L., and Abdallah, H. B. (2012). Literature review of reuse in business process modeling. *Software & Systems Modeling*.

Hallerbach, A., Bauer, T., and Reichert, M. (2010). Capturing variability in business process models: The provop approach. *Journal of Software Maintenance and Evolution: Research and Practice*, 22(6-7):519–546.

Indulska, M., Green, P. F., Recker, J., and Rosemann, M. (2009). Business process modeling: Perceived benefits. In Laender, A. H. F., Castano, S., Dayal, U., Casati, F., and de Oliveira, J. P. M., editors, *ER*, volume 5829 of *Lecture Notes in Computer Science*, pages 458–471. Springer.

Jin, Tao, J. W. W. N., Rosa, M. L. R., and ter Hofstede, A. H. M. (2010). Efficient and accurate retrieval of business process models through indexing - (short paper). In Meersman, R., Dillon, T. S., and Herrero, P., editors, *OTM Conferences (1)*, volume 6426 of *Lecture Notes in Computer Science*, pages 402–409. Springer.

Jin, T., Jianmin, W., Nianhua, W., Marcello, L. R., and ter Hofstede, A. H. M. (2013). Efficient querying of large process model repositories. *Computers in Industry*, 64(1):41–49.

Jin, T., Wang, J., and Wen, L. (2011). Querying business process models based on semantics. In *DASFAA (2)*, pages 164–178.

Keller, G. and Teufel, T. (1998). *Sap R/3 Process-Oriented Implementation: Iterative Process Prototyping*. SAP databases. Addison Wesley Professional.

La Rosa, M., Dumas, M., Uba, R., and Dijkman, R. (2013). Business process model merging: An approach to business process consolidation. *ACM Trans. Softw. Eng. Methodol.*, 22(2):11:1–11:42.

Makni, L., Haddar, N. Z., and Ben-Abdallah, H. (2011). Semantic design patterns for business processes. In Cuaresma, M. J. E., Shishkov, B., and Cordeiro, J., editors, *ICSOFT (1)*, pages 83–87. SciTePress.

Makni, L., Haddar, N. Z., and Ben-Abdallah, H. (2012). Detection of semantic relations between business process activity labels. In *ICEIS (3)*, pages 273–277.

Makni, L., Khlif, W., Haddar, N. Z., and Ben-Abdallah, H. (2010). A tool for evaluationg the quality of business process models. In *ISSS/BPSC*, pages 230–242.

Nicola, A. D., Missikoff, M., Proietti, M., and Smith, F. (2010). An open platform for business process modeling and verification. In *DEXA (1)*, pages 76–90.

OMG (2013). Bpmn 2.0. http://www.omg.org/spec/BPMN/2.0.2/.

Oryx-Editor (2010). Web-based graphical business process editor. http:// code.google.com/p/ oryx-editor/.

Rosa, M. L. (2009). *Managing variability in process-aware information systems*. PhD thesis, Queensland University of Technology.

Rosa, M. L., Dumas, M., and ter Hofstede, A. H. (2009). Modelling business process variability for design-time configuration. In Cardoso, J. and van der Aalst, W. M., editors, *Handbook of Research on Business Process Modeling*, pages 204–228. Information Science Reference (IGI Global), Hershey, PA. Access to the author-version is currently restricted pending permission from the publisher. For more information, please refer to the publisher's website (see hypertext link) or contact the author.

Rosa, M. L., Lux, J., Seidel, S., Dumas, M., and ter Hofstede, A. H. M. (2007). Questionnaire-driven configuration of reference process models. In Krogstie, J., Opdahl, A. L., and Sindre, G., editors, *CAiSE*, volume 4495 of *Lecture Notes in Computer Science*, pages 424–438. Springer.

Rosemann, M. and van der Aalst, W. M. P. (2007). A configurable reference modelling language. *Inf. Syst.*, 32(1):1–23.

Scheidegger, C. E., Vo, H. T., Koop, D., Freire, J., and Silva, C. T. (2008). Querying and re-using workflows with vstrails. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1251–1254, New York, NY, USA. ACM.

Shao, Q., Sun, P., and Chen, Y. (2009). Wise: A workflow information search engine. In Ioannidis, Y. E., Lee, D. L., and Ng, R. T., editors, *ICDE*, pages 1491–1494. IEEE.

ter Hofstede, A. H. M., Ouyang, C., Rosa, M. L., Song, L., Wang, J., and Polyvyanyy, A. (2013). Apql: A process-model query language. In *AP-BPM*, pages 23–38.

Weber, B., Reichert, M., and Rinderle-Ma, S. (2008). Change patterns and change support features - enhancing flexibility in process-aware information systems. *Data Knowl. Eng.*, 66(3):438–466.