Applying Patterns during Business Process Modeling*

Thomas Gschwind[1], Jana Koehler[1], and Janette Wong[2]

[1] IBM Zurich Research Laboratory,
Switzerland
www.zurich.ibm.com/csc/bit
[2] IBM Software Group, Canada

Abstract. Although the business process community has put a major emphasis on patterns, notably the famous workflow patterns, only limited support for using patterns in today's business process modeling tools can be found. While the basic workflow patterns for control flow are available in almost every business process modeling tool, there is no support for the user in correctly applying these simple patterns leading to many incorrectly modeled business processes. Only limited support for pattern compounds can be found in some tools, there is no active support for selecting patterns that are applicable in some user-determined context, tools do not give feedback to the user if applying a pattern can lead to a modeling error, nor do they trace the sequence of applied patterns during the editing process.

In this paper, we describe an extension of a business process modeling tool with patterns to provide these capabilities. We distinguish three scenarios of pattern application and discuss a set of pattern compounds that are based on the basic workflow patterns for control flow. We present an approach where business users receive help in understanding the context and consequences of applying a pattern.

1 Introduction

There is wide agreement that patterns can accelerate the process of designing a solution and reduce modeling time, while at the same time they enable an organization to more easily adopt best practices [1,2,3]. Patterns enable participants of a community to communicate more effectively, with greater conciseness and less ambiguity. Furthermore, process patterns are considered as an effective means to bridge the Business IT gap. Bridging this gap is more critical than ever because IT advances have escalated the rate of development of new business functions and operations [2].

Despite the common belief in the importance of patterns, only limited support for using patterns in today's business process modeling tools can be found. While the basic workflow patterns for control flow [4] are available in most business process modeling tools and the YAWL system [5] provides all workflow patterns, applying even a basic pattern is under the full responsibility of the user. It is thus not surprising that most modeling errors result from incorrect combinations of the *exclusive choice*, *parallel split*, *simple merge*, and *synchronization* patterns [6].

* The work published in this article was partially supported by the SUPER project (http://www.ip-super.org/) under the EU 6th Framework Programme Information Society Technologies Objective (contract no. FP6-026850).

M. Dumas, M. Reichert, and M.-C. Shan (Eds.): BPM 2008, LNCS 5240, pp. 4–19, 2008.
© Springer-Verlag Berlin Heidelberg 2008

In this paper, we discuss flexible pattern support where users can apply patterns to unstructured process models, they obtain active support in selecting patterns that are applicable in some user-determined context, the tool gives feedback to the user if applying a pattern can lead to a modeling error and it traces the sequence of applied patterns during the editing process. We focus on the basic workflow patterns for control flow, because of their frequent usage during business process modeling and discuss a set of pattern compounds that can be built from them. We present an infrastructure that automates parts of the pattern application process. The infrastructure analyzes the consequences of applying a pattern with respect to the soundness of the resulting process model and enables only those patterns that are correctly applicable in a given context, which we describe using the category of the *process fragments* to which the pattern is applied. Information about the fragment category is obtained from the *process structure tree* that results from parsing the workflow graph underlying the process model [7,8].

We show how patterns can be integrated in a modeling tool such that they enable business users to move away from a drawing tool with drag-and-drop editing capabilities to a true business-level process modeling tool that allows users to arrive at models of higher quality with less effort. Although we only consider control-flow patterns, we see our contribution as an important prerequisite for extending powerful pattern support to more concrete business process patterns that describe best practices, because these patterns will usually contain control-flow information as one essential part of the pattern description [3]. We do not yet introduce a domain-specific vocabulary for the control-flow patterns, but we argue that it is necessary to do so in the future to make the patterns more easily usable by business users.

The paper is organized as follows: In Section 2 we revisit the basic workflow patterns for control flow and define three scenarios for the application of control-flow patterns during an iterative process modeling approach: (1) refinement of a *single* control-flow edge by applying a block-oriented pattern compound, (2) application of a pattern compound to a *pair* of selected control-flow edges, (3) application of a basic pattern to a *set* of selected control-flow edges. Sections 3, 4, and 5 present the three scenarios of pattern application in more detail. Section 3 also provides details on our infrastructure that is based on the process structure tree [7] and that enables us to extend the application of patterns to unstructured process models. Section 6 summarizes initial experiences with an implementation of the three pattern scenarios in a commercial business process modeling tool. The section also discusses the interplay of process patterns, process refactoring operations, and process model transformation. Section 7 gives an overview on the state of the art in business process patterns, while Section 8 concludes the paper.

2 The Workflow Patterns Revisited

When talking about business process patterns, many business process experts refer to the famous workflow patterns [4] that have their origin in comparing the runtime constructs available in existing workflow engines. Figure 1 shows the most widely used subset of the control-flow patterns. We selected these patterns to build active pattern support into a business process modeling tool.

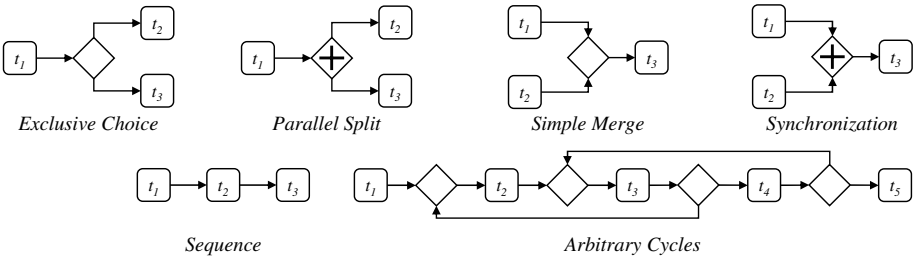

Fig. 1. The five basic workflow patterns *exclusive choice*, *parallel split*, *simple merge*, *synchronization* and *sequence*. In addition, the *arbitrary cycles* pattern as the most frequently occurring pattern for iteration [4].

The patterns as shown in Figure 1 are of course available in most business process modeling tools in the form of gateway icons that business users can drag and drop on a canvas and connect to other modeling elements. Unfortunately, this availability of the patterns in today's modeling tools is insufficient to enable users to successfully reuse proven solutions to recurring problems. The workflow patterns are too fine-grained and not sufficiently enriched with information on the context and consequences to represent a reusable solution. A possible alternative, as for example implemented in the ADEPT2 system [9,10], is to offer block-structured *pattern compounds* and *change patterns* that allow users to model structured workflows by an editing process where processes are sound by construction.

In this paper, we are especially interested in pattern-support for the editing of unstructured process models where the soundness of these models is not guaranteed by construction. We developed a pattern-based modeling prototype by extending the commercial modeling tool IBM WebSphere Business Modeler with *pattern compounds* that we built from the basic control-flow patterns. Our special emphasis is on *pattern sequences*, i.e., how a model unfolds pattern by pattern and how a user creates an unstructured model by applying patterns in an iterative and tool-supported modeling process.

The process models that we consider are generalizations of workflow graphs that permit multiple start and end events. Following [11] we define them as follows:

A business process model is a directed graph $G = (N, E)$ where each node $n \in N$ is either a start or end event, an activity, or a gateway with the gateways partitioned into the types exclusive choice, parallel split, simple merge, and synchronization, satisfying the following conditions

1. there is at least one start event and at least one end event; each start event has no incoming edges and exactly one outgoing edge, whereas each end event has exactly one incoming edge but no outgoing edges,
2. the exclusive choice and parallel split have exactly one incoming edge and two or more outgoing edges, whereas the simple merge and synchronization have two or more incoming edges and exactly one outgoing edge; each activity has exactly one incoming and exactly one outgoing edge,
3. the graph is connected and each node $n \in N$ is on a path from a start to an end event.

We are adopting BPMN notation to draw the process models and pattern structures. This means that we use a diamond to depict a gateway and in the case of a parallel split or synchronization, a plus sign is added to the diamond. Activities are depicted with rounded corner rectangles, while a start event is depicted with an empty circle and an end event is depicted with a thick circle.

Table 1 gives an overview of three *pattern application scenarios* that we discuss in this paper. Each scenario is applicable to process models that are still unfinished, i.e., they may not fully comply to the definition above.

Table 1. Overview of pattern application scenarios

Sc.	**selected process elements**	**applied pattern compound**	**source**	**target**
1	single edge	well-formed sound block	sound	sound
2	pair of edges	gateway-guarded control flow	sound	sound/unsound
3	set of edges	gateway	sound/unsound	sound/unsound

In Scenario 1, a user selects a *single edge* in a model. This edge is replaced by a pattern compound that represents a well-formed process fragment. The user has four choices of pattern compounds that he can apply: *sequence*, *parallel compound*, *alternative compound*, and *cyclic compound*. This form of pattern application (sometimes also denoted as transition refinement) is always possible in our tool. When applied to a sound process model or fragment thereof, it preserves the soundness, i.e., it cannot introduce any modeling errors. Section 3 discusses this scenario in more detail.

By *soundness* of a process model, we mean the absence of *deadlocks* and *lack of synchronization*. In other words, no situation occurs where some part of the process is waiting indefinitely for another part of the process and no part of the process executes more often than intended because of two tokens that occur on the same edge. A formal account of soundness would go beyond the scope of this paper, but can be found in [12,7].

In Scenario 2, a user selects a *pair of edges* in the model to which he can add a new gateway-guarded control flow. Two pattern compounds are available to the user which we denote as *alternative branch* and *parallel branch*. This scenario allows the user to also create arbitrary cycles. The pattern application is always possible, but an unshielded application can introduce new modeling errors, i.e., a process or fragment with a sound underlying workflow graph can become unsound. We describe this scenario in Section 4 and discuss how potential soundness problems can be discovered and prevented.

In Scenario 3, the user selects a *set of edges* to redirect existing control flow such that it starts or ends in a newly introduced gateway. In this scenario, the basic control-flow patterns are directly available to the user. They can be applied to any process model or fragment thereof and either maintain soundness, yield an unsound model or correct an unsound model into a sound one. In Section 5 we describe an infrastructure that alerts the user of these situations and thereby extends the limited support for basic workflow patterns that is available today.

Our scenarios differ by the user-triggered selection of modeling elements and by the class of process models that the user can create with the patterns that are available for

each selection. The focus on the selection of modeling elements is important to help business users understand how to apply a pattern. Furthermore, it provides them with a simple and systematic description of the context of a selected pattern in the form of the surrounding process fragment, and the consequences in terms of soundness of the resulting model, while the modeling tool exploits this information to automate the pattern application. We believe that a higher degree of automation is essential because we are addressing non-technical users in contrast to software developers who traditionally apply software patterns in a mostly manual process.

3 Scenario 1: Applying Patterns to a Single Edge

Our first scenario has been widely studied by the workflow community, e.g., as a form of transition refinement [13]. We introduce it here in order to review some essential prerequisites for structured workflow modeling that we then gradually relax in Scenarios 2 and 3. Scenario 1 provides the user with the most simple form of application of a pattern where he can select a single control-flow edge to further refine the business process model. Instead of selecting a single edge, the user can also select a single activity in the process model. In this case, our tool assumes that with this selection, the single outgoing control-flow edge of this activity is selected, i.e., the pattern is applied following the activity in the control flow.[1]

In this scenario, we provide users with *pattern compounds* that represent a well-formed and sound block-structured fragment of a process. These pattern compounds have been studied within the context of structured workflows [14,15,16,17] and are also available in ADEPT2 [9,10]. Four types of block-structured pattern compounds are available to the user:

- *sequence*: a totally ordered set of connected activities,
- *parallel compound*: a parallel split followed by a synchronization that are connected by two or more branches containing one or more activities[2],
- *alternative compound*: an exclusive choice followed by a simple merge,
- *cyclic compound:* a simple merge followed by an exclusive choice.

Figure 2 illustrates this mode of pattern application, which restricts the user to model structured workflows, but which are guaranteed to be sound by construction. The initial sequence of activities in this example can be either created manually or by using the *sequence* pattern. Alternatively, we offer an *auto-link transformation* where the user only places the activities that he wants to be part of the initial sequence in an approximate horizontal arrangement. Then he invokes the auto-link transformation that takes a set of horizontally arranged activities and produces a fully connected sequential process model including a start and an end event.

Aalst [14] and Kiepuszewski et al. [15] showed that only a subset of all workflow graphs can be generated when using block-structured process fragments. However,

[1] We do not consider here the refinement of a single activity into a subprocess, which is a completely different scenario.

[2] The number of branches and the names of activities can be provided as parameters when invoking the pattern.

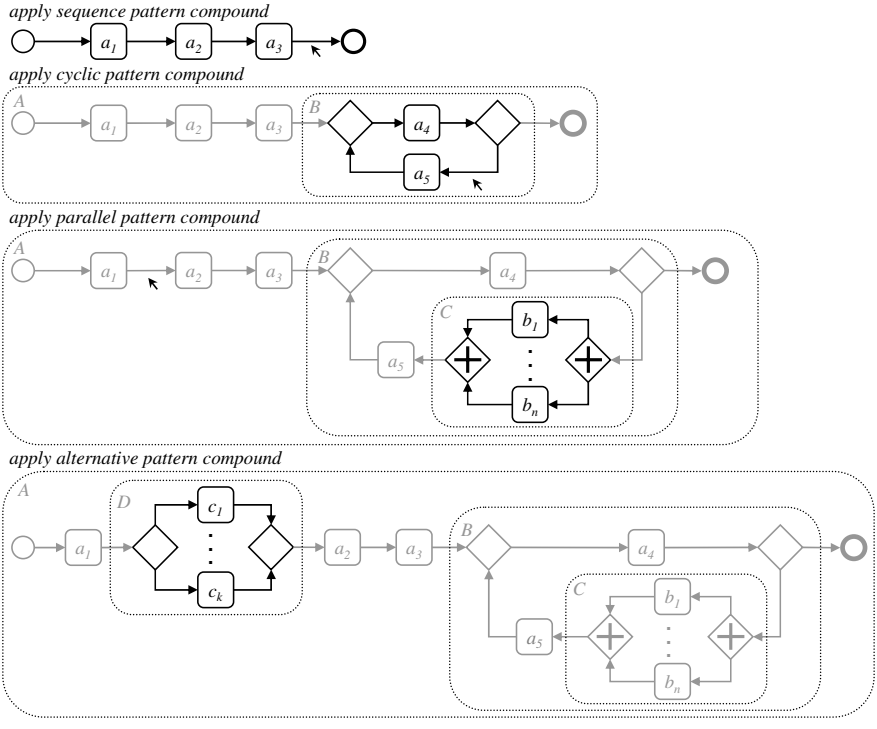

Fig. 2. Sound refinement of a process model by applying block-structured pattern compounds to a single edge

modeling block-structured processes is practically relevant for two reasons: First, these models are more comprehensible to human users [18]. Secondly, they can be directly mapped to structured process execution languages such as BPEL and thus make it much easier to go from business to IT.

In order to trace the successive application of patterns, i.e., the pattern sequence, and to determine the context under which a pattern can be correctly applied, we use the *process structure tree* (PST) [7,8] and the notion of *category* of a fragment. In Scenario 1, illustrated by Figure 2, the PST is used to trace the successive application of patterns by the user. Scenarios 2 and 3 will illustrate how the PST together with the category notion can be used to help a user correctly apply patterns to unstructured process models.

The PST results from parsing the workflow graph of the process model. It represents a unique decomposition of a workflow graph into canonical *SESE fragments*, which are either disjoint or fully nested in each other. A SESE fragment is a non-empty subgraph of the workflow graph that is bordered by a single-entry and a single-exit (SESE) edge [7] or node [8]. The dotted rounded-corner rectangles in Figure 2 show the SESE edge fragments of the example. Note that only maximal sequences are canonical fragments, e.g., the sequence containing activity a_1 followed by fragment D is not a

canonical fragment, because fragments a_2, a_3, and B are part of the same sequence. The PST can be computed in linear time. It is unique and modular, i.e., a local change of the workflow graph only causes a local change of the decomposition. It is as fine-grained as possible when using SESE node fragments [8].

To determine whether a pattern can be correctly applied, the category of a fragment is important, which is defined by syntactic properties of the underlying workflow graph. Well-structured, acyclic concurrent, unstructured alternative, and complex fragments were proposed in [7]. Other categorizations can be defined instead, i.e., we define and use our own categories *sequence*, *alternative branching* (an arbitrary number of XOR-splits and XOR-joins that must be cycle-free), *parallel branching* (an arbitrary number of AND-splits and AND-joins that must be cycle-free), and *cyclic alternative branching*, which is an alternative branching that is not cycle-free. Fragments in these categories are known to be sound. Figure 3 illustrates the categories *parallel branching* and *cyclic alternative branching* with two unstructured example models.

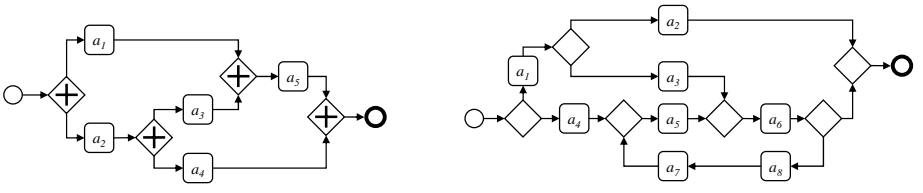

Fig. 3. Fragment categories parallel branching (left) and cyclic alternative branching (right)

Figure 4 shows the PST for the example of Figure 2 based on SESE edge fragments.

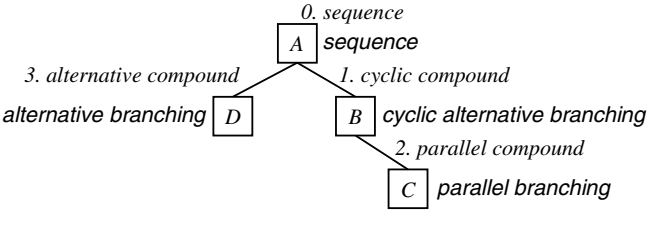

Fig. 4. The process structure tree

The nodes of the PST, which represent the fragments, are annotated with the category of the fragment. The edges are annotated with a number and pattern name providing us with the history of pattern application, i.e., the pattern sequence that the user applied in this example: 0. sequence, 1. cyclic compound, 2. parallel compound, 3. alternative compound. Applying patterns in Scenario 1 adds new fragments to the tree. When applying the patterns of Scenario 2 and 3, fragments and their category can change locally in the PST, e.g., a fragment can also disappear.

4 Scenario 2: Applying Patterns to a Pair of Edges

As refining process models with block-structured patterns is very limiting for many business modeling scenarios, we now consider a first generalization where the user selects an ordered pair of edges (s, t). The first selected edge s is considered the *source* of the new flow and the second selected edge t is considered its *target*. The user can select any two edges as source and target edges.[3]

In this scenario, we support two pattern compounds *alternative branch* and *parallel branch* that the user can apply to establish a new control flow between the source and the target. We provide the pattern compounds in the form of gateway-guarded control-flow edges:

- *alternative branch*: an exclusive choice with a single outgoing edge leading to a simple merge,
- *parallel branch*: a parallel split with a single outgoing edge leading to a synchronization.

Figures 5 and 6 illustrate a typical example of this pattern application scenario. In Figure 5 we see part of a mortgage approval process with two alternative branches. If the customer is not creditworthy, a rejection is sent by the bank and the application by the customer is closed. If the customer is creditworthy, a mortgage offer is sent, the documents are completed, and an account is set up for paying out the mortgage and the mortgage is registered.

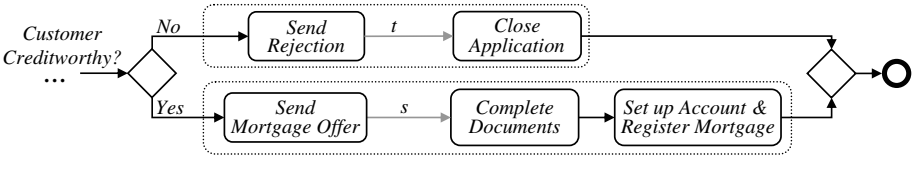

Fig. 5. Example of a simple mortgage approval process

When taking a closer look at this process model, we notice that it assumes that the customer accepts the mortgage offered by the bank. However, this may not always be the case. If the offer is rejected by the customer, the bank employee should contact the customer to find out why and then also close the application. To achieve this change in the process model, the user selects the edges s and t as the source and target and applies the alternative branch pattern. The result can be seen in Figure 6. In a parameterized version of this pattern, a list of activities can be provided that is placed on the newly added branch.

In this example, the user transforms a structured model into an unstructured, but sound model. If the user had applied the parallel branch pattern compound to Figure 5, a deadlock error would have been introduced. In order to prevent such situations, knowledge of the fragment categories maintained within the PST is essential when patterns

[3] Alternatively, a user can select two activities where the tool takes the outgoing edge of the first activity as the source and the incoming edge of the second activity as the target.

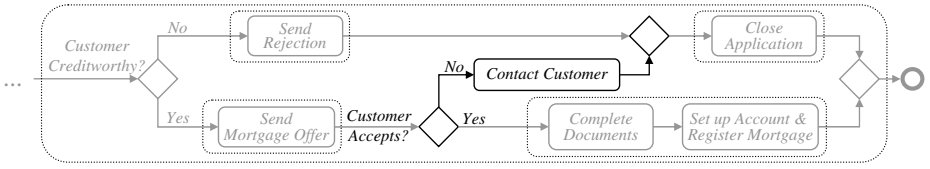

Fig. 6. Adding an alternative branch pattern to two selected edges

are applied to a pair of edges. In this example, the user selects two edges that belong to two different fragments of type sequence that each comprise one of the decision branches for the *Customer Creditworthy?* decision. The pattern application destroys these fragments. They are replaced by four smaller fragments of type sequence—two on each branch. Their parent fragment, which spawns the process fragment of Figure 5, remains unchanged and also preserves its type *acyclic sequential branching*.

The tool guides the user in applying the pattern compounds by analyzing the SESE fragments that contain the selected edges. If the selected pair of edges is a pair of entry/exit edges of a SESE fragment, all pattern compounds that we discussed for Scenarios 1 and 2 are applicable independently of the category of the fragment. If the user selects a pair of edges where at least one of the edges is not an entry or exit edge of a fragment, an analysis of the SESE fragments surrounding the selected edges needs to be performed. The tool analyzes the SESE fragments containing the source and target edge and all those SESE fragments that enclose these fragments up to the smallest SESE fragment that contains both edges. All fragments have to be of the same category, which decides if a parallel or alternative branch can be applied to the edges, see Table 2.

Table 2. Soundness of pattern application based on fragment category

Fragment category	adding cycle allowed?	parallel branch	alternative branch
sequence	yes	×	√
sequence	no	√	×
cyclic alternative branching	yes	×	√
parallel branching	no	√	×

Rows 1 and 2 show that if the fragment categories are a simple sequence of activities, both branch patterns are applicable to a selected pair of edges. However, only the alternative branch pattern is permitted to add a cycle to the process model (row 1). Adding a parallel branch such that a cycle is introduced would lead to a deadlock and is thus not permitted (row 2). The (acyclic) alternative branching is a special case of the cyclic alternative branching and is thus subsumed by row 3. Adding an alternative branch to a parallel branching fragment is not permitted, because it would introduce a lack of synchronization error (row 4).

If the selected edges do not satisfy the conditions with respect to the fragment categories or if the process model is known to be unsound, patterns can still be applied in our current prototype, because we do not want to constrain the user too much in

using the pattern-based editing capability. However, a warning, but no further guidance is given to the user. Note that it is not possible to eliminate a deadlock or lack of synchronization error by only applying one of the six pattern compounds that we discussed so far.

5 Scenario 3: Applying Patterns to a Set of Edges

In our last scenario, we consider the most general situation where the user has selected a set of two or more edges or nodes. This means, the selections possible in Scenario 2 can occur here as well, but we consider application scenarios for the basic patterns *parallel split*, *synchronization*, *exclusive choice*, and *simple merge*. When applying the basic patterns to unsound fragments, it is possible for the user to correct modeling errors.

We want to support users selecting nodes in addition to selecting edges because in the midst of editing a process model, it is common to encounter nodes without connecting edges yet and applying patterns to nodes can be very useful to complete the editing. On the other hand, it is not usual to have dangling edges without nodes, because business process modeling tools make users add nodes first and then allow them to connect the nodes with edges. We support three situations if nodes are selected:

1. all selected nodes have incoming edges, but no outgoing edges,
2. all selected nodes have outgoing edges, but no incoming edges,
3. all nodes are fully disconnected, i.e., have neither incoming nor outgoing edges.

In situation (1), a new outgoing edge is added to each node and the synchronization or simple merge pattern is enabled depending on the fragment type returned for the selection. In situation (2), a new incoming edge is added to each node and the parallel split or exclusive choice pattern is enabled. In situation (3), all four basic patterns are enabled and depending on the selection of a pattern by the user, either a new outgoing or incoming edge is added to each node.

Currently, we impose very restrictive constraints when applying the basic patterns to a selection of nodes or edges. For example, a synchronization pattern can be added if a parallel split is found in the process model from which all selected nodes or edges can be reached without encountering other non-AND gateways along the path. Similar conditions can be formulated for the other three basic patterns.

Figure 7 illustrates an example situation. In case that the user only selects activities a_3, a_4, and a_5 in the process model shown in the left, the exclusive choice is found that can only be correctly matched with a simple merge. If in addition, a_1 is selected as well, the parallel split is found, but on three of the four paths, the exclusive choice is encountered. In such a situation, more than one pattern must be applied as is shown in the right of the figure, which requires refactoring techniques that are subject of our ongoing work [19]. The same challenges occur when the user selects a set of edges. Again, we constrain the pattern application as described above. In addition, we have to consider the nodes that are connected by the selected edges.

Figure 8 illustrates an example process where the user wants to introduce two join points in the process flow. With his first selection e_1, e_2, e_3, the user wants to join the three parallel flows to allow the doctor to talk to the patient after having worked out the

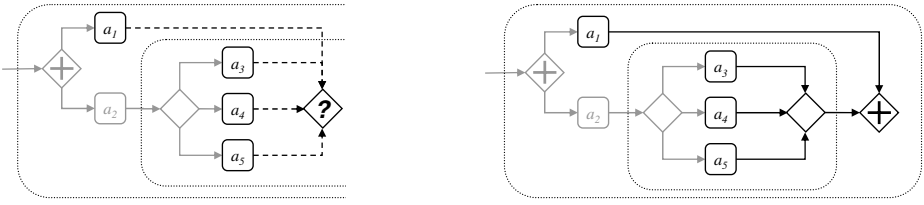

Fig. 7. Applying a single basic pattern to merge or synchronize all selected nodes/edges is not possible without introducing an error into the process model. Instead, two basic patterns must be applied.

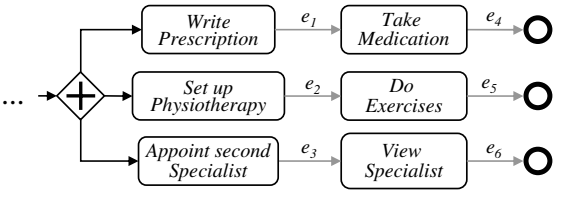

Fig. 8. A medical example process with two edge selections e_1, e_2, e_3 and e_4, e_5, e_6

patient's prescriptions. With his second selection e_4, e_5, e_6, the user wants to join the individual process ends and allow the doctor to review the treatment results with the patient.

Figure 9 shows the result of applying two synchronization patterns to the example, each parameterized with an activity that follows the added AND-Join.

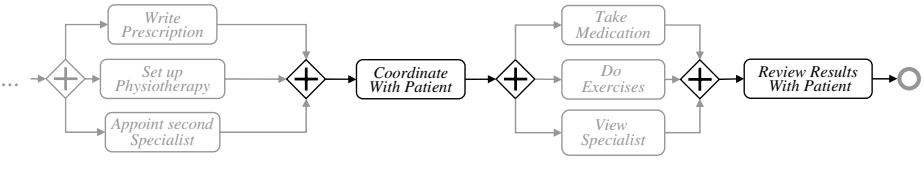

Fig. 9. Selecting a set of edges for joining leads to a fusion of nodes into a single node when they are identical. In the case of different nodes, applying a pattern to join branches also requires applying another pattern to split the branches again, because the nodes cannot be merged.

Applying the synchronization pattern to the first selection e_1, e_2, e_3 requires the introduction of an additional AND-Split following the AND-Join resulting from the pattern, because the subsequent activities cannot be merged. In the case of gateways, a merging is sometimes possible, but it may not always be desired by the user. We take a conservative approach so far and do not merge gateways. When applying the synchronization pattern to the second selection e_4, e_5, e_6, the end events can be merged into a single node and no additional gateway is needed. After having applied the two patterns, the process can be further improved by for example applying the cyclic pattern compound to iterate the prescription and treatment for the patient if necessary.

6 Implementation and Validation of Pattern-Based Editing

Figure 10 shows a screen capture of our prototype implementation where we added pattern support in the form of additional plug-ins to IBM WebSphere Business Modeler, which is an Eclipse-based commercial business process modeling tool. As not all process models can be generated with the patterns and pattern compounds that we described in this paper, we also provide the user with refactoring and transformation operations in addition to the normal editing capabilities.

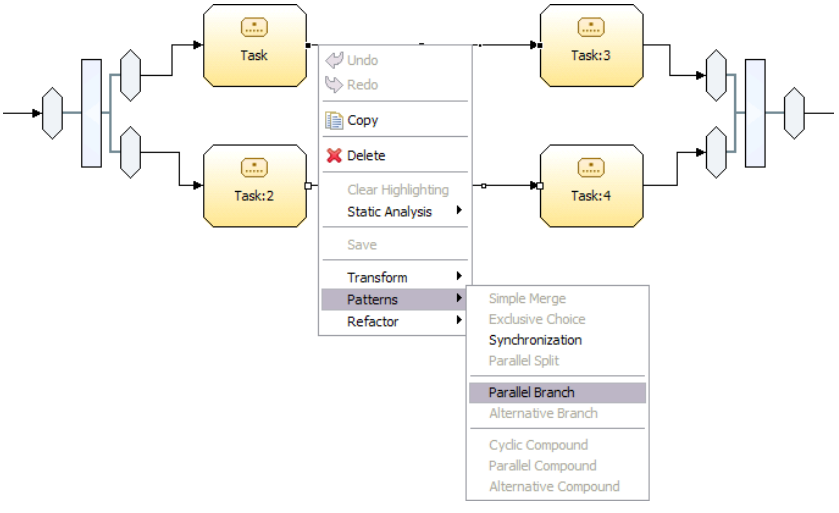

Fig. 10. Context-sensitive pattern availability when selecting a pair of edges within a parallel branching fragment

Our set of currently implemented transformations has been described in [20] and includes a first prototype of a transformation that introduces data flow. We plan to extend this set of transformations by adopting and extending transformations that have been described in [14,16,17,21]. Several of these transformations require a tool to verify the soundness of the process model to which they are applied or that they create. With the linkage of our transformations and patterns to the PST and its fragments, we have laid the foundation to perform these checks much faster. In many practical cases a process model only contains simply structured fragments where soundness can be decided in linear time. For the general case, we currently develop a complete soundness checker that we can can invoke on complex fragments. The feedback to the user about potential modeling errors that can be introduced into a model when applying a pattern is clearly valuable to increase the quality of the process models.

One can easily demonstrate that by applying patterns, transformations, and refactoring operations on a business process model, many time-consuming editing operations can be replaced by a single click [20]. A first adoption of the plug-ins by IBM consultants showed that on average about 10% of the modeling time can be saved with

up to 70% of the pure editing time. About 50% of all users who installed the plug-ins use them frequently in their daily work. Approx. 10% find them very easy to use, while two-thirds said that they need practice. 90% of all users confirmed that the plug-ins help them in improving the quality of their models.

Adding pattern-based support for data flow was the most frequently requested extension of the plug-ins. When applied to a single edge, patterns can inherit the data flow from the single edge. If several edges are selected that carry different data items, many possible ways to resolve such a situation exist that we currently explore. Our approach can also be extended to activities that have multiple incoming and outgoing edges, but then requires different disambiguation techniques to determine the edges to which a pattern must be applied in case the user selects one or more activities. As such a disambiguation is not always possible, slightly modified patterns with more constrained application conditions must be developed.

7 Related Work

A growing divide in the patterns world is discussed that opens between the pattern experts who continue to document patterns and the pattern users who are rarely aware of relevant patterns and understand how to leverage and apply them [22]. Only little adoption of patterns by practitioners is observed leading to a rather low impact of the pattern experts on the expected pattern users. The reason for the low adoption of patterns is located in the difficulty to find, contextualize, and compose patterns. “To use a cooking analogy, what they find is a list of ingredients when what they really want is a recipe” [22]. This observation is more than true for the business user working with a business process modeling tool today.

In order to enable users to adopt and actively use patterns, tools must allow users to build applications by progressively applying patterns. However, how and if patterns can be built into tools is a hot debate [23]. Following the pioneering work by Gamma et al. [24], patterns must be thoroughly described by the commonly recurring *problem*, the *context* and *consequences* of applying the pattern, and the *solution* provided by the pattern itself. Understanding the context and consequences related to a specific pattern is a very important human-centered task. Tools that help users in achieving this task must provide active support to select patterns and apply them in composition steps towards creating a complete solution for a particular scenario. The challenge is that “tools that work with patterns would have to be able to semantically understand your design as well as the pattern’s trade-offs” [23]. By linking pattern application to the process structure tree, its fragment categories and their soundness, we have built an initial semantic understanding into our business process modeling tool.

Process patterns are found at three levels of abstraction [3]: (1) *abstract process patterns* that capture generic process structures applicable to any process, (2) *general process patterns* that capture proven process elements applicable across different domains, (3) *concrete process patterns* that capture hands-on experience and best practices for recurring business functions in a specific domain. The most prominent example of abstract patterns are probably the famous workflow patterns for control flow [4], which have also been complemented by patterns for data flow and resources. Examples

of general process patterns are discussed in [25,26]. A famous collection of concrete process patterns is [27]. Further examples of concrete process patterns are discussed in [28,29,30]. All three levels of process patterns can be built into a business process modeling tool. While it is argued that many abstract patterns can provide significant opportunities for reuse [3]—hence our initial focus on the basic control-flow patterns—it is also emphasized that patterns should be presented in the domain vocabulary of the business user [31] for easier recognition and application, which we have not addressed yet.

Three building blocks of a process pattern-based approach are proposed that must be built into a tool [3]: (1) a *pattern inventory*, (2) support for *pattern selection* and (3) *pattern realization*. Only the ADEPT2 system [9,10], with which we coincide on Scenario 1, seems to implement solutions for the pattern inventory and the selection and realization phases. Some abstract use cases have also been formulated: they comprise the listing, insertion, connection, visualization, and removal of patterns [30] of which we address the context-sensitive listing and the correct insertion of a pattern in a process model in this paper. Palettes that group patterns for specific purposes are discussed in [29,30] and a concrete design of such a palette is shown in [20].

A significant part of research is devoted to pattern languages. An example of such a pattern language for processes implemented in a service-oriented architecture is discussed in [25], while [32,33] describe a visual pattern language for the representation and enforcement of quality constraints in process models. UML-based metamodels and pattern languages are proposed in [34,35,28]. An application of domain-specific modeling languages for the IT-oriented refinement of business processes is discussed in [36]. In this paper, we do not focus on a specific representation of patterns in some language or metamodel, but present an initial collection of patterns for specific scenarios that we found useful for business users.

Unfortunately, very few practical recommendations for the reuse of process patterns are given to business users. Havey [37] emphasizes the need for high quality, but gives only two very simplistic and not so easy to follow recommendations: keep a process model to a size that it fits on a single page (if necessary by using subprocesses) and model in a coarse-grained way, i.e., focus on the main process activities. Our active guidance of the user in applying a pattern helps us to go beyond approaches of syntax-based editing [38] that constrain editing operations by syntactic properties of a model, because we focus on a linkage to a semantic analysis addressing soundness.

8 Conclusion

In this paper, we present three different scenarios of pattern application in a business process modeling tool. For each scenario, we discuss a set of patterns and pattern compounds that are linked to an effective structural and semantic analysis of the business process model based on its process structure tree in order to guide the user in applying a pattern. This analysis helps business users in understanding the context and consequences of applying a pattern and enables the tool to actively support a user during pattern selection and application. Future work will focus on developing a comprehensive

set of patterns, refactoring operations and model transformations for the most frequent use cases in business process modeling including a refinement of process models with data flow.

References

1. Buschmann, F., Henney, K., Schmidt, D.: Past, present and future trends in software patterns. IEEE Software 24(7/8), 31–37 (2007)
2. Medicke, J., McDavid, D.: Patterns for business process modeling. Business Integration Journal 1, 32–35 (2004)
3. Tran, H., Coulette, B., Thuy, D.: Broadening the use of process patterns for modeling processes. In: Proc. SEKE, Knowledge Systems Institute Graduate School, pp. 57–62 (2007)
4. van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow patterns. Distributed and Parallel Databases 14(1), 5–51 (2003)
5. van der Aalst, W., ter Hofstede, A.: Yawl: Yet another workflow language. Information Systems 30(4), 245–275 (2005)
6. Koehler, J., Vanhatalo, J.: Process anti-patterns: How to avoid the common traps of business process modeling. IBM WebSphere Developer Technical Journal 10(2+4) (2007)
7. Vanhatalo, J., Völzer, H., Leymann, F.: Faster and more focused control-flow analysis for business process models though SESE decomposition. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 43–55. Springer, Heidelberg (2007)
8. Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) Proc. BPM 2008. LNCS, vol. 5240, pp. 100–115. Springer, Heidelberg (2008)
9. Reichert, M., Rinderle, S., Kreher, U., Dadam, P.: Adaptive process management with ADEPT2. In: 21st Int. Conference on Data Engineering, pp. 1113–1114. IEEE, Los Alamitos (2005)
10. Weber, B., Rinderle, S., Reichert, M.: Change patterns and change support features in process-aware information systems. In: Krogstie, J., Opdahl, A., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 574–588. Springer, Heidelberg (2007)
11. Sadiq, W., Orlowska, M.: Analyzing process models using graph reduction techniques. Information Systems 25(2), 117–134 (2000)
12. van der Aalst, W., Hirnschall, A., Verbeek, H.: An alternative way to analyze workflow graphs. In: Pidduck, A.B., Mylopoulos, J., Woo, C.C., Ozsu, M.T. (eds.) CAiSE 2002. LNCS, vol. 2348, pp. 535–552. Springer, Heidelberg (2002)
13. van der Aalst, W., van Hee, K.: Workflow Management: Models, Methods, and Systems. MIT Press, Cambridge (2002)
14. van der Aalst, W.: Verification of workflow nets. In: Azéma, P., Balbo, G. (eds.) ICATPN 1997. LNCS, vol. 1248, pp. 407–426. Springer, Heidelberg (1997)
15. Kiepuszewski, B., ter Hofstede, A., Bussler, C.: On structured workflow modeling. In: Wangler, B., Bergman, L.D. (eds.) CAiSE 2000. LNCS, vol. 1789, pp. 431–445. Springer, Heidelberg (2000)
16. Sadiq, W.: On business process model transformations. In: Laender, A.H.F., Liddle, S.W., Storey, V.C. (eds.) ER 2000. LNCS, vol. 1920, pp. 267–280. Springer, Heidelberg (2000)
17. Eder, J., Gruber, W., Pichler, H.: Transforming workflow graphs. In: Proc. INTEROP-ESA, pp. 203–216. Springer, Heidelberg (2006)
18. Mendling, J., Reijers, H., Cardoso, J.: What makes process models understandable? In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 48–63. Springer, Heidelberg (2007)

19. Vanhatalo, J., Völzer, J., Moser, S., Leymann, F.: Automatic workflow graph refactoring and completion (submitted for publication)
20. Koehler, J., Gschwind, T., Küster, J., Pautasso, C., Ryndina, K., Vanhatalo, J., Völzer, H.: Combining quality assurance and model transformations in business-driven development. In: Proc. AGTIVE. LNCS, vol. 5088. Springer, Heidelberg (2008)
21. van der Aalst, W., Basten, T.: Inheritance of workflows: An approach to tackling problems related to change. Theoretical Computer Science 270(1-2), 125–203 (2002)
22. Manolescu, D., Kozaczynski, W., Miller, A., Hogg, J.: The growing divide in the patterns world. IEEE Software 24(4), 61–67 (2007)
23. Kircher, M., Völter, M.: Software patterns. IEEE Software 24(7/8), 28–30 (2007)
24. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading (1994)
25. Zdun, U., Dustdar, S.: Model-driven and pattern-based integration of process-driven SOA models. Int. Journal of Business Process Integration and Management 2(2), 109–119 (2007)
26. Barros, O.: Business information system design based on process patterns and frameworks. BPTrends 9, 1–5 (2004)
27. Malone, T., Crowston, K., Herman, G.: Organizing Business Knowledge: The MIT Process Handbook. MIT Press, Cambridge (2003)
28. Tran, H., Coulette, B., Thuy, D.: A UML-based process meta-model integrating a rigorous process patterns definition. In: Münch, J., Vierimaa, M. (eds.) PROFES 2006. LNCS, vol. 4034, pp. 429–434. Springer, Heidelberg (2006)
29. Thom, L., Iochpe, C., Reichert, M.: Workflow patterns for business process modeling. In: 8th Workshop on Business Process Modeling, Development, and Support in conjunction with CAISE 2007 (2007)
30. Thom, L., Lau, J., Iochpe, C., Mendling, J.: Extending business process modeling tools with workflow pattern reuse. In: Proc. ICEIS 2006. LNBIP, vol. 3, pp. 447–452. Springer, Heidelberg (2007)
31. Rising, L.: Understanding the power of abstraction in patterns. IEEE Software 24(7/8), 46–51 (2007)
32. Förster, A., Engels, G., Schattkowsky, T., Straeten, R.: A pattern-driven development process for quality standard-conforming business process models. In: IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2006), pp. 135–142. IEEE, Los Alamitos (2006)
33. Förster, A., Engels, G., Schattkowsky, T., Straeten, R.: Verification of business process quality constraints based on visual process patterns. In: Proc. TASE, pp. 197–208. IEEE, Los Alamitos (2007)
34. Störrle, H.: Describing process patterns with UML. In: Ambriola, V. (ed.) EWSPT 2001. LNCS, vol. 2077, pp. 173–181. Springer, Heidelberg (2001)
35. Hagen, M., Gruhn, V.: Process patterns - a means to describe processes in a flexible way. In: Proc. ProSim (2004), `http://prosim.pdx.edu/prosim2004`
36. Brahe, S., Bordbar, B.: A pattern-based approach to business process modeling and implementation in web services. In: Georgakopoulos, D., Ritter, N., Benatallah, B., Zirpins, C., Feuerlicht, G., Schoenherr, M., Motahari-Nezhad, H.R. (eds.) ICSOC 2006. LNCS, vol. 4652, pp. 166–177. Springer, Heidelberg (2007)
37. Havey, M.: Essential Business Process Modeling. O'Reilly, Sebastopol (2005)
38. Ehrig, H., Engels, G., Kreowski, H.J., Rozenberg, G.: Handbook of Graph Grammars and Computing by Graph Transformation, vol. 2. World Scientific, Singapore (1999)