CCBR–Driven Business Process Evolution

Barbara Weber[1], Stefanie Rinderle[2], Werner Wild[3], and Manfred Reichert[4]

[1] Quality Engineering Research Group, Institute of Computer Science,
University of Innsbruck – Technikerstrasse 21a, 6020 Innsbruck, Austria
Barbara.Weber@uibk.ac.at
[2] Dept. Databases and Information Systems, University of Ulm, Germany
rinderle@informatik.uni-ulm.de
[3] Evolution Consulting, Innsbruck, Austria
werner.wild@evolution.at
[4] Information Systems Group, University of Twente, The Netherlands
m.u.reichert@cs.utwente.nl

Abstract. Process-aware information systems (PAIS) allow coordinating the execution of business processes by providing the right tasks to the right people at the right time. In order to support a broad spectrum of business processes, PAIS must be flexible at run-time. Ad-hoc deviations from the predefined process schema as well as the quick adaptation of the process schema itself due to changes of the underlying business processes must be supported. This paper presents an integrated approach combining the concepts and methods provided by the process management systems ADEPT and CBRFlow. Integrating these two systems enables ad-hoc modifications of single process instances, the memorization of these modifications using conversational case-based reasoning, and their reuse in similar future situations. In addition, potential process type changes can be derived from cases when similar ad-hoc modifications at the process instance level occur frequently.

1 Introduction

For a variety of reasons companies are developing a growing interest in aligning their information systems in a process-oriented way to provide the right tasks to the right people at the right point in time. However, when automating business processes it is extremely important not to restrict users. Early attempts to realize process-aware information systems (PAIS) have been unsuccessful whenever rigidity came with them [1,2]. Therefore, a flexible PAIS must allow authorized users to deviate from the pre-modeled process schema if needed (e.g., by dynamically inserting, deleting or moving process steps). In addition, the PAIS must be quickly adaptable to changes of the underlying business processes, e.g., due to business process reengineering efforts or the introduction of new laws [3,4,5].

In the ADEPT project we have developed a next generation process management system (PMS) that satisfies these needs. On the one hand, the ADEPT PMS offers full functionality with respect to the modeling, analysis, execution,

H. Muñoz-Avila and F. Ricci (Eds.): ICCBR 2005, LNCS 3620, pp. 610–624, 2005.

and monitoring of business processes [1,3,6]. On the other hand, it provides support for adaptive processes at both the process instance and the process type level. Changes at the instance level may affect single process instances and be performed in an ad-hoc manner, e.g., to deal with exceptional or unanticipated situations [1]. Process type changes, in turn, can be applied to adapt the PAIS to business process changes. In this context, concurrent migration of hundreds up to thousands of process instances to the new process schema may become necessary. ADEPT allows to perform the respective migrations on-the-fly while preserving process consistency and system robustness [3,6,7].

In practice, process type changes are often driven by previous ad-hoc adaptations of individual process instances. Usually, similar or equivalent changes of a larger number of process instances indicate the need for adapting the process type (i.e., the process template) itself [8]. For example, in a patient treatment process an additional lab test activity has been inserted for a significant number of process instances; in order to better reflect the real-world process, a process schema evolution should then be initiated to create a new process template version which includes this additional activity (cf. Fig. 2). So far, ADEPT has not adequately dealt with this fact and has not considered the reuse of information about previous ad-hoc changes. In particular, it has not maintained semantic information about these changes (e.g., their reason and context). Thus, it has been the responsibility of the process designer to identify frequently applied changes and to adapt process types accordingly.

By contrast, CBRFlow [9] enables users to apply process instance changes in a more intelligent way. Particularly, it allows to document the reasons for a process instance change and to reuse information about previously performed ad–hoc changes when defining new ones. For this conversational case-based reasoning (CCBR) [10] is used. So far, focus has been put on ad–hoc changes of single process instances whereas process type changes have not yet been considered. In order to provide comprehensive change support a PAIS must capture the whole *process life cycle* and all kinds of changes in an integrated way.

In this paper we provide such an integrated approach, which combines the concepts and methods offered by ADEPT and CBRFlow: On the one hand, the combined system provides a powerful process engine, which supports all kinds of changes in one system. On the other hand, it enables the intelligent reuse of process instance changes and the derivation of process type changes from the collected information. The added value offered by this integration is shown in Table 1.

Table 1. Benefits from Integrating ADEPT and CBRFlow

	ADEPT	CBRFlow	ADEPT+CBRFlow
process instance changes	+	+	+
reuse of process instance changes		+	+
process type changes	+		+
deriving process type changes			+

Section 2 provides background information, Section 3 discusses issues that arise when trying to derive process type changes from cases. In addition to the resulting evolution of the business processes the corresponding case-bases evolve over time as well. This important issue is covered in Section 4. Section 5 discusses related work and Section 6 closes with a summary and an outlook on future work.

2 Background

In this section we provide background information regarding process management and case-based reasoning (CBR) as used in our approach.

2.1 Process Management

For each business process supported (e.g., booking of a business trip or handling a medical order) a *process type* T has to be defined. Formally, such a type is represented by a *process schema* S of which different versions may exist. In Fig. 1, for example, S and S' correspond to different schema versions of the same process type T (thus reflecting the evolution of T).

In the following, a process schema is represented by a directed graph, which defines a set of *activities* – the process steps – and the control flow between them.[1] In Fig. 1 process schema S consists of 6 activities: for example, activity `Admit patient` is followed by activity `Make appointment` in the flow of control whereas `Prepare Patient` and `Inform Patient` can be processed in parallel. Formally:

Definition 1 (Process Schema). *A process schema S is defined by a tuple (N, E) where N denotes the set of activities and E the set of control edges (i.e., precedence relations) between these activities.*

At runtime new process instances can be created and executed based on schema S. Similar to Petri Nets, the execution state of a particular process instance is captured by a marking function $M = (NS, ES)$. It assigns to each activity n its current status $NS(n) \in$ {NOT_ACTIVATED, ACTIVATED, FINISHED} and to each control edge its marking $ES(e) \in$ {NOT_SIGNALED, SIGNALED}. For the top most process instance $I_\nu^{(1)}$ in Fig. 1, for example, activity `Admit patient` has already been finished and therefore its outgoing edge is marked as SIGNALED. Activity `Make appointment`, in turn, is currently activated, i.e., offered to users for execution in their worklists.

Usually, a process instance I is executed according to the control flow defined by its original schema S. As motivated in Section 1, however, users may have to deviate from the original schema (e.g., by adding new activities or by

[1] In this paper we restrict our considerations to schemes with sequential and parallel activities. Our approach, however, considers more complex control structures as well (e.g., conditional branchings, loops, and synchronizations between parallel execution branches). Details of the process meta model used can be found in [1,6,7].

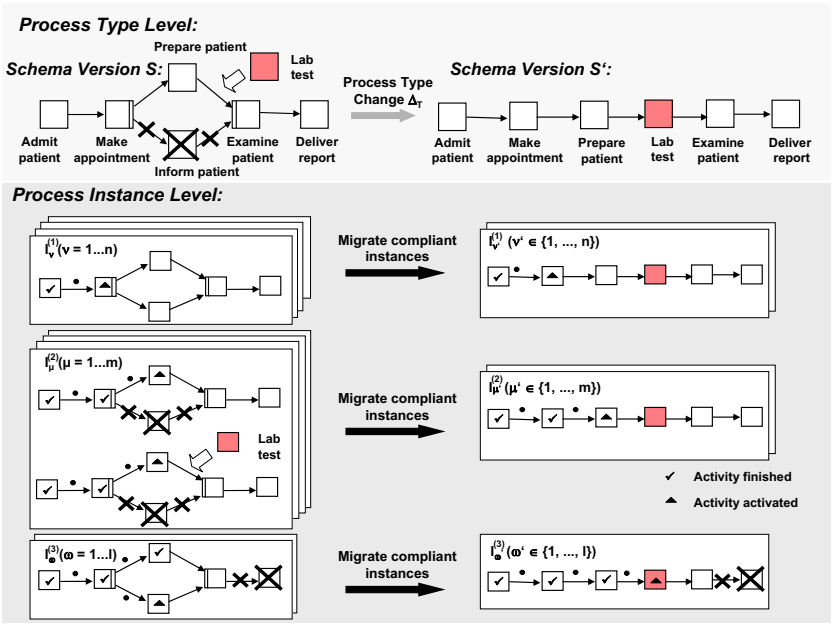

Fig. 1. Migration of Process Instances – Clinical Example

deleting existing ones). For this reason, we must distinguish between two basic classes of process instances, those that still follow their original schema and those that have been individually modified during runtime. In the following, we call instances of the former class *unbiased* and those of the latter one *biased*. Correspondingly, a biased instance I cannot solely be characterized by its original schema S and marking M, but must also capture the sequence of ad-hoc changes $\Delta_I = (a_1, \ldots, a_k)$ applied to it so far. Generally, several ad-hoc changes may have been applied to a biased instance I at different points in time.

For example, consider Fig. 1: Process instances $I_\nu^{(1)}, \nu = 1 \ldots n$ are unbiased. By contrast, process instances $I_\mu^{(2)}$, $\mu = 1 \ldots m$ and $I_\omega^{(3)}$, $\omega = 1 \ldots l$ are biased since their current execution schema deviates from their original schema S. Instances $I_\omega^{(3)}$, $\omega = 1 \ldots l$, for example, are biased due to the dynamic deletion of activity `Deliver report`. Formally:

Definition 2 (Process Instance).
A process instance I is defined by a tuple (S, Δ_I, M) where

- *$S = (N, E)$ denotes the process schema I was originally created on.*
- *$\Delta_I = (a_1, \ldots, a_k)$ comprises the instance–specific sequence of ad–hoc modifications which have been applied to I so far (i.e., changes transforming the process schema S, instance I was created from, into the current execution schema $S_I = S + \Delta_I = (N', E')$). Thereby $a_i = (op, s, paramList)$ denotes an operation $op \in OP$ which operates on a schema subject s (i.e., activities*

Table 2. *A Selection of ADEPT Change Operations*[*]

Change Operation *op* **applied to Schema S**	**Effects on Schema S**
Additive Change Operations	
serialInsert(S, X, A, B)	insert activity X into schema S between the two directly connected activities A and B
parallelInsert(S, X, (A))	insert activity X into schema S parallel to activity A
Subtractive Change Operations	
deleteActivity(S, X)	delete activity X from schema S

[*]A detailed description of all change operations supported by ADEPT can be found in [11,12].

or edges) using parameters paramList. OP is the set of change operations provided by ADEPT, a subset of these operations is given in Table 2.

- *M =(NS, ES) reflects the current marking of I. It assigns to each activity* $n \in N'$ *its current status NS(n) and to each edge* $e \in E'$ *its marking ES(e).*

2.2 Case-Based Reasoning and Learning Processes

Case-based reasoning is a contemporary approach to problem solving and learning. New problems are dealt with by applying past experiences – described as cases – and by adapting their solutions to the new problem situation [13]. Thus, CBR contributes to incremental and sustained learning: Every time a new problem is solved, information about it and its solution is retained and therefore immediately made available for solving future problems [14].

Conversational CBR is an extension to the CBR paradigm, which actively involves users in the inference process [15]. A CCBR system can be characterized as an interactive system that, via a mixed-initiative dialogue, guides users through a question-answering sequence in a case retrieval context. Unlike traditional CBR, CCBR does not require the user to provide a complete a priori problem specification for case retrieval, nor requires him to provide knowledge about the relevance of each feature for problem solving. Instead, the system assists the user in finding relevant cases by presenting a set of questions to assess the given situation. Furthermore, it guides users who may supply already known information on their initiative. Therefore, CCBR is especially suitable for handling exceptional or unanticipated situations that cannot be dealt with in a fully automated way.

In our approach a case c represents a concrete ad-hoc modification of a process instance I which can be reused by other instances. It consists of a textual problem description, a set of question-answer pairs, and a solution part (i.e., the action list). The question–answer pairs describe the reasons for the ad-hoc change and the action list comprises the change operations (and related context information) applied to I.

Definition 3 (Case, Case–Base).
A case c is a tuple (pd, { q_1an_1, ..., q_nan_n }, sol, freq) where

- *pd is a textual problem description*
- *{ q_1an_1, ..., q_nan_n} denotes a set of question-answer pairs*
- *sol = { a_j | $a_j = (op_j, s_j, paramList_j)$, j = 1, ..., k} is the solution part of the case denoting a list of actions (i.e., a set of changes that have been applied to one or more process instances; see also Def. 2)*
- *freq ∈ ℕ denotes the reuse frequency of case c*

A case–base CB = { $c_1, \ldots, c_m$} is defined as a set of cases.

3 Deriving Evolutionary Process Changes from Cases

Fig. 2 illustrates our approach: it shows how CCBR is used to perform ad-hoc changes of single process instances (cf. Section 3.1) and how it triggers process type changes if the same or similar ad-hoc changes happen over and over again (with respect to instances of a given process type; cf. Section 3.2). Fig. 2 also indicates that the evolution of a process schema may require the concurrent migration of the associated case-base (cf. Section 4).

As already mentioned, new instances can be created based on a given process schema and then be executed according to that schema. If required, authorized users may deviate from the pre-modeled process schema during runtime at the level of single process instances. They apply CCBR to retrieve knowledge about previous ad-hoc changes. In addition, they document the new change and collect information about the reasons which required the respective ad-hoc deviation. This information is then immediately available for future reuse in similar situations. Finally, if a case is frequently reused (i.e., the same ad-hoc change is often applied to instances of a particular process type), case usage may exceed a predefined threshold. In this situation, the knowledge engineer is notified about the potential need of a process type change. He can then take action, e.g., by adapting the process type and migrating the case-base.

3.1 Performing Ad-Hoc Changes Using CCBR

Integrating ADEPT and CBRFlow offers promising perspectives: It allows for ad-hoc modifications at the process instance level in a correct and consistent manner, it facilitates the memorization of these modifications using CBR techniques, and it provides for reusing respective cases in similar, future situations. The underlying CBR cycle [14] can be described as follows:

Adding a New Case. Whenever a user wants to apply an ad-hoc change at the process instance level and no similar cases can be found in the CCBR system, she adds a new case $c = (pd, \{q_1an_1, \ldots,\}$, sol, 1) to the case-base. The user enters this case by briefly describing the current problem, by entering a set of question-answer pairs describing the reasons for the ad-hoc deviation, and by specifying the actions to be taken from the list of available change operations.

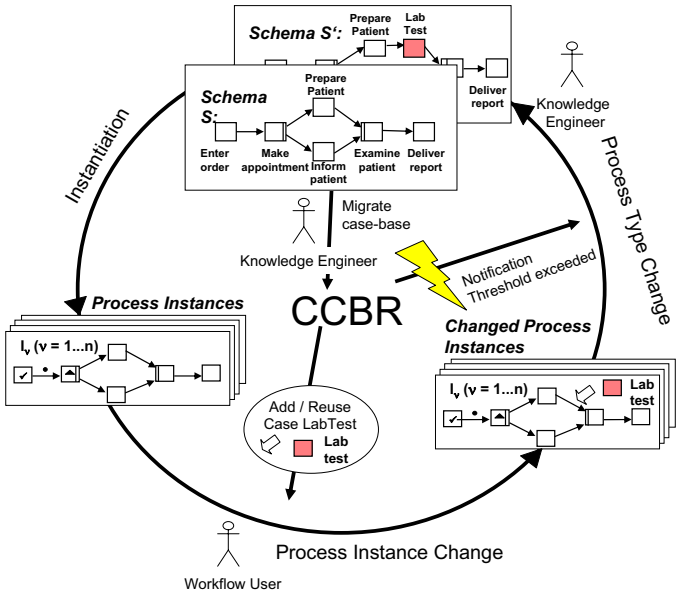

Fig. 2. Deriving Evolutionary Process Changes from Cases

Question-answer pairs can be entered by either selecting the question from a list of previously entered questions (i.e., reusing questions from existing cases) or, when no suitable question is already in the system, by defining a new question and giving the appropriate answer. Depending on the permissions of the user and the current state of the process instance (i.e., which activities are currently performed) only a subset of the ADEPT operations may be applicable. The user selects the desired change operations $op_1, \ldots, op_p$ and the subjects $s_1, \ldots, s_p$ they operate on (e.g., activities and control edges). In addition, she provides the parameters for each selected operation. Finally, the case is retained and thus immediately made available for future reuse.

Retaining a Case. Unlike CBRFlow [9], our approach stores cases not relative to the location in the process graph where the ad-hoc modification occurred (e.g., relative to an activity), but in reference to the process schema itself. There is one case-base version for each process schema version S, as they might be relevant at different locations in the process. For example, the insertion of a particular activity (e.g., order lab test) might become necessary at different points in time during process execution.

Case Retrieval. For case retrieval the CCBR approach as described in [10] has been adapted. When deviations from the predefined process schema become necessary the user initiates case retrieval in the CCBR component. The system then assists her in finding already stored, similar cases by presenting a set of questions. Users can directly answer any of the displayed questions (in arbitrary order) or additionally apply a filter to the case-base by specifying an operation *op*

as well as the subject s on which the operation is supposed to operate. Filtering is done by selecting values from predefined lists and by ignoring those cases that do no match the filter criteria (i.e., that do not have the selected operation and subject in the actions list); only the remaining cases are presented. Formally:

Definition 4 (Filtered Case–Base). *Let $CB = \{c_1, \ldots, c_k\}$ be a case–base with $c_i = (\ldots, sol_i, \ldots)$ $(i = 1, .., k)$ and $sol_i = \{(op_j, s_j, \ldots)\}$ $(j = 1, .., m)$ (cf. Def. 3). Then the filtered case-base CB_{filter} can be determined as follows:*

$$CB_{filter} = \begin{cases} \{c_i \in CB \mid \exists (op_j, s_j, \ldots) \in sol_i : op_j = op \wedge s_j = s\} & if A \\ \{c_i \in CB \mid \exists (op_j, s_j, \ldots) \in sol_i : op_j = op\} & if B \\ CB & otherwise \end{cases}$$

whereby

- *A: user has specified change operation $op \in OP$ and subject s*
- *B: user has specified change operation $op \in OP$*

The system then searches for similar cases by calculating the similarity for each case in the case-base CB_{filter}. It then displays the top n ranked cases (ordered by decreasing similarity) and their reputation score, which indicates how successfully each case has been applied in the past. Similarity is calculated by dividing the number of correctly answered questions minus the number of incorrectly answered questions by the total number of questions in the case. Formally:

Definition 5 (Similarity). *Let c = (pd^c, QA^c= $\{q_1^c an_1^c, \ldots, q_n^c an_n^c\}$, ...) be a case of case–base CB and $Q = \{q_1^Q an_1^Q, \ldots, q_m^Q an_m^Q\}$ be a query against CB. Then sim(Q, c) denotes the similarity between Q and c. Formally:*

$$sim(Q,c) = \frac{same(Q,c) - diff(Q,c)}{|QA^c|}$$

whereby

- *same(Q, c)* $= |QA^c \cap Q|$
- *diff(Q, c)* $= |\{q_i^c an_i^c \in QA^c \mid \exists\ q_j^Q an_j^Q \in Q$ *with* $q_i^c = q_j^Q \wedge an_i^c \neq an_j^Q$; $i = 1,..,n$; $j = 1,..,m\}|$

Case Reuse. ADEPT supports different kinds of ad-hoc changes which, for example, allow users to skip activities, to change activity orders, or to insert new activities [1]. In particular, the system ensures that ad-hoc changes do not lead to unstable system behavior[2] or to inconsistent instance states. When an exceptional or unexpected situation occurs, the user is assisted in selecting the desired change operations and in setting the change context (e.g., the predecessors and successors of an activity to be inserted) accordingly.

Generally, change definition requires user experience, in particular if the intended change requires concurrent adaptations (e.g., when deleting a particular

[2] None of the guarantees (e.g., absence of deadlocks, correctness of data flow) which have been achieved by formal checks at buildtime are violated due to the change.

activity, data-dependent activities may have to be deleted as well). Therefore, the reuse of existing knowledge about previous ad-hoc changes is highly desirable. When a user decides to reuse an existing case, the actions specified in the solution part of the case are forwarded to and carried out by the ADEPT change engine. The reuse counter is increased and a work item is created for evaluating the ad-hoc change later on to maintain the quality of the case-base.

When the reuse counter exceeds a certain configurable threshold the knowledge engineer is notified about the potential need to perform a schema evolution (cf. Section 3.2). Altogether, the reuse of existing ad-hoc changes contributes to hide as much complexity from users as possible.

Ensuring Quality Through Case Evaluation. The accuracy of the cases in the case-base is crucial for the overall performance of a CBR system and consequently for the trust users have in it. When cases are not added by the knowledge engineer but by end users, evaluation mechanisms are needed to ensure quality of the cases in the case-base.

Therefore, similar to Cheetham and Price [16], we propose to augment the CBR cycle with the ability to determine the confidence in the accuracy of individual solutions. However, for CCBR systems the accuracy cannot be determined automatically as the semantics of the question-answer pairs are, unlike in traditional CBR systems, unknown to the system. For this purpose we apply the concept of reputation from e-commerce where such systems are used to build trust among strangers like, for instance, in eBay's feedback forum [17]. There, each positive feedback on a transaction increases the reputation score of a seller, while each negative feedback results in a decrease. In our approach, we use the concept of reputation to indicate how successfully a case has been reused in the past, i.e., how much it has contributed to the performance of the case-base, thus indicating the degree of confidence regarding the accuracy of this case. Like in eBay, users are encouraged to provide feedback when adding or reusing a case. For this purpose, a new work item representing an optional feedback task is automatically created and inserted into the worklist of the user who entered or applied the case. She can then rate the performance of the case either with 1 (positive), 0 (neutral) or -1 (negative), and may optionally specify an additional comment. The reputation score of a case is then calculated as the number of distinct users who gave a positive feedback minus the number of those who gave a negative feedback. Negative feedback usually results in a notification of the knowledge engineer (see below).

During case retrieval the CCBR system displays the overall reputation score together with a table of the totals of each rating in the past 7 days, the past month, and the past 6 months to the user. Upon request the user can read all comments provided in the past and decide whether the reputation of the case is high enough for her to have confidence in its accuracy.

Case Revision. Negative feedback results in a notification of the knowledge engineer who can then revise the case or decide to deactivate it (no deletion is allowed to foster traceability).

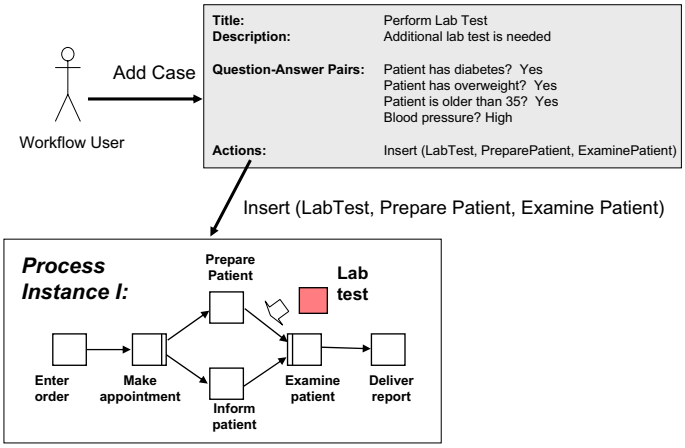

Fig. 3. Adding a New Case to Insert a Process Step

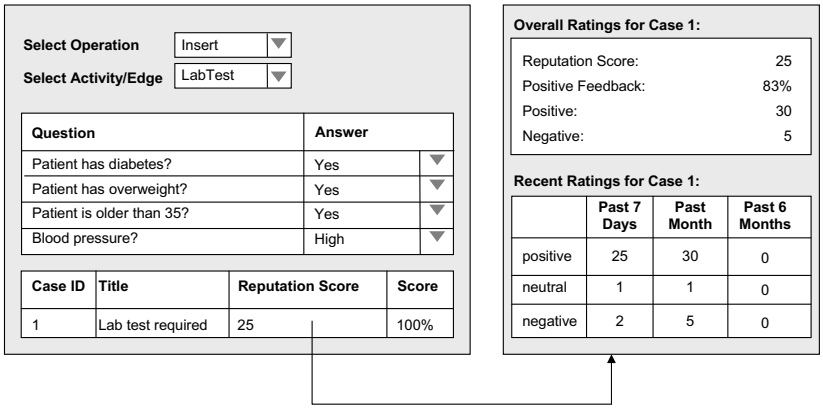

Fig. 4. Retrieving Similar Cases

Example. To illustrate the above concepts we provide a simplified medical example. As depicted in Fig. 1 the examination of a patient usually takes place after a preparation step. During the examination the physician recognizes that the patient suffers from diabetes and he detects several other important risk factors. Therefore, the physician decides to request an additional lab test for the patient to be performed after activity `Prepare patient` and before activity `Examine Patient`. As the system contains no similar cases, the physician enters a new case describing the situation and the action to be taken (Fig. 3). ADEPT then checks whether the insertion of activity `Lab Test` is possible for the respective process instance, and - if so - applies the specified insert operation to that instance. The latter includes updating the instance markings and user worklists. If, for example, `Prepare patient` is completed and `Examine Patient`

is activated, this activation will be undone (i.e., respective work items are removed from user worklists) and the newly inserted activity `Lab test` becomes immediately activated. In any case, the newly inserted activity is treated like the other process steps, i.e., the same scheduling and monitoring facilities exist.

When talking with another diabetic patient some time later, the physician remembers that there has been a similar situation before and initiates the CCBR sub-system to retrieve similar cases. As he still remembers that he had performed an additional lab test, he selects the `Insert` operation as well as the `Lab Test` activity to filter the case-base. He then answers the questions presented by the system, finds the previously added case, and reuses it (Fig. 4). Of course, the physician could also directly answer any of the presented questions without selecting an operation or an activity first (e.g., when he doesn't remember a similar previous situation).

3.2 Deriving Process Type Changes

When the usage of a particular case exceeds the specified threshold value (i.e., based on the frequency the case was reused, cf. Def. 3), the system sends a notification to the knowledge engineer. He may then initiate a process type change in order to derive a new version of the process schema. For this purpose he may directly apply the change operations captured by the respective case; alternatively, he can adapt the case's operation set (e.g., by only considering a subset of it).

When a new process schema is released future instances can be created from it. However, the challenging question is how to treat already running process instances, i.e., instances that have been derived from the old process schema version. Particularly for long-running processes, it is crucial that respective instances can be migrated to the new process schema version if desired (cf. Fig. 1). In this context ADEPT first checks whether these instances are *compliant* with the new process schema or not. Compliant means that the process schema change can be applied to the instance in its current state so that it can be smoothly re–linked to the new schema, i.e., *migrated* to it without causing inconsistencies or errors (e.g., deadlocks). Then the set of compliant process instances is divided into unbiased and biased instances. The former can be directly re–linked to the new schema. For each instance its marking with respect to the new schema version is automatically determined. For biased process instances further correctness checks are necessary, e.g., regarding structural correctness (for details see [12]). Finally, all compliant process instances are running according to the new schema version whereas non compliant process instances remain running on the old schema. An example is given in Section 4.

4 Migrating the Case-Base

Assume that the frequencies for reusing certain cases exceed specified thresholds (cf. Section 3.2). For instance, as illustrated in Fig. 5 the specified thresholds for reusing case c1 ($freq = 51$) and c5 ($freq = 60$) are exceeded, thus triggering a

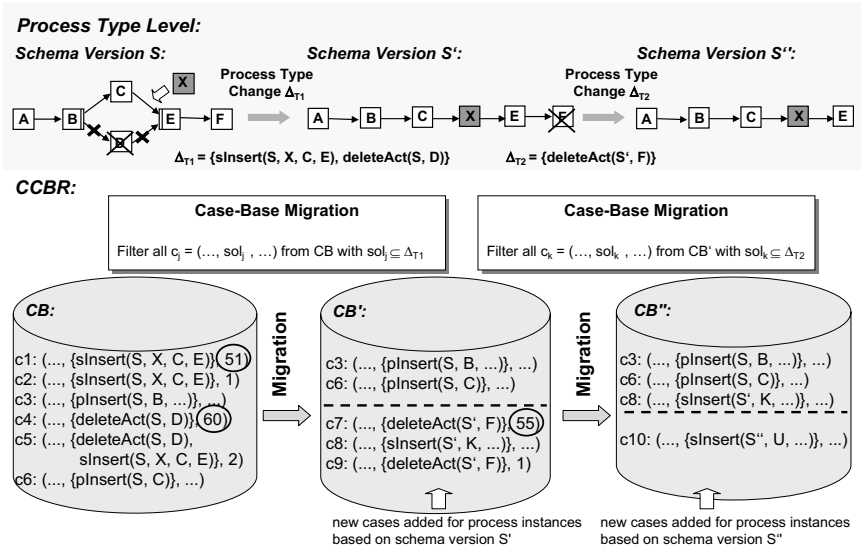

Fig. 5. Migrating the Case-Base

process type change. The knowledge engineer is informed and decides that the respective instance changes serialInsert(S,X,C,E) (sInsert(S,X,C,E) for short) and deleteActivity(S,D) (deleteAct(S,D) for short) should be pulled up to the process type level. He derives a new process schema version S' by applying process type change Δ_{T1} = {sInsert(S,X,C,E), deleteAct(S,D)}.

This process type change is accompanied by the migration of compliant process instances to the new schema version S', whereas non-compliant process instances remain running on the old schema version (cf. Section 3.2). In addition, the challenging question is, which cases of the previous case-base CB (on S) shall be valid for process instances of S' as well. This consideration becomes necessary as the solution part of certain cases may be covered by a process type change Δ_T. Therefore the respective cases are no longer needed. In our approach, only cases whose solution part is not reflected in the process type change Δ_T are migrated to CB'. By contrast, cases whose solution part is a subset of Δ_T are omitted. Formally:

Definition 6 (Case-Base Migration). *Let $CB = (c_1, \ldots, c_k)$ be a case-base stored for process instances running according to process schema S. If then process type change Δ_T transforms S into another process schema S' the new version CB' of CB can determined as follows:*

$$CB' = CB \setminus \{c_i = (\ldots, sol_j, \ldots) \in CB \mid sol_j \subseteq \Delta_T \ (j = 1,..,m)\}$$

In the example depicted by Fig. 5, cases c1 and c4 that initiated the process type change, as well as case c2 and c5 are already covered by the new schema version S'. Consequently, the new version CB' of case-base CB is built by mi-

grating only cases c3 and c6. Of course, new cases may be added to CB' due to ongoing ad-hoc changes of instances based on S'. Again, the migration of this case-base will become necessary if another process schema migration takes place later on. In our example, type change Δ_{T2} = {deleteAct(S',F)} is triggered by case c7 which exceeds a certain frequency $freq$ (55). The resulting case-base CB" is shown in Fig. 5.

5 Related Work

This paper is based on the idea of integrating PMS and CCBR. In related work CBR has been applied to support process modeling [18,19], to the configuration of complex core processes [20], to the handling of exceptions [21] and for the composition of Web Services [22]. All of these approaches apply traditional CBR, to our knowledge there are no other approaches relying on CCBR.

Related work also includes adaptive process management. Existing approaches either support ad-hoc changes at the process instance level or schema modifications at the process type level (for an overview see [3]). Except for ADEPT [12] none of these approaches considers both kinds of changes in an integrated manner. In particular the full life cycle support using CCBR techniques has not been addressed so far. Though CBRFlow [9] fosters the reuse of ad-hoc changes, it has not yet considered process type changes. This gap is closed by the integration of ADEPT and CBRFlow.

AI planning, especially mixed-initiative case-based planning (e.g., NaCoDAE/HTN [23], MI-CBP [24], SiN [25] and HICAP [26]) can be seen as complementary to our approach as we primarily focus on the execution of processes and not on modeling or planning. Process management approaches rely on a predefined process schema (i.e., plan) that is instantiated during run-time in high numbers. In contrast, in AI planning the user is supported in generating a new plan for every new problem situation, which prevents the problem of having to change other running instances of the same plan. Other than in AI planning our meta-model supports complex control flow constructs (e.g., conditional branching, loop backs, and synchronizations between parallel execution branches).

Process-based knowledge management systems are suitable for knowledge intensive workflows and are often used to provide additional process information to the user in order to support them during the execution of activities (e.g., DECOR [27], FRODO TaskMan [28], KnowMore [29]). FRODO TaskMan extends the approach taken in KnowMore by supporting integrated modeling and enactment of weak workflows. Like our approach, FRODO TaskMan allows instance level modifications of the workflow during run-time, but does not support process type changes. Additionally it supports working with an incomplete process schema due to its late modeling capabilities.

6 Summary and Outlook

The integration of ADEPT and CBRFlow offers promising perspectives. It results in a new generation of adaptive process technology, which facilitates and

speeds up the implementation of new as well as the adaptation of existing processes. Both, the capability to quickly and correctly propagate type changes to in-progress process instances as well as the intelligent support of ad-hoc adaptations will be key ingredients in next generation PMS, resulting in highly adaptive PAIS. Currently, we are working on the implementation of a prototype that combines the methods and concepts provided by ADEPT and CBRFlow. Future research will include the evaluation of this approach in different application settings, like healthcare processes and emergent workflows (e.g., in the automotive domain). Our future research will include the extension of the presented approach towards agile process mining, i.e., fostering to start with a simple, incomplete process schema and then learn from the living processes to evolve the schema over time.

References

1. Reichert, M., Dadam, P.: ADEPT$_{flex}$ - supporting dynamic changes of workflows without losing control. JIIS **10** (1998) 93–129
2. Jørgensen, H.D.: Interactive Process Models. PhD thesis, Norwegian University of Science and Technology, Trondheim, Norway (2004)
3. Rinderle, S., Reichert, M., Dadam, P.: Correctness criteria for dynamic changes in workflow systems – a survey. Data and Knowledge Engineering **50** (2004) 9–34
4. Casati, F., Ceri, S., Pernici, B., Pozzi, G.: Workflow evolution. Data and Knowledge Engineering **24** (1998) 211–238
5. v.d. Aalst, W., Basten, T.: Inheritance of workflows: An approach to tackling problems related to change. Theoret. Comp. Science **270** (2002) 125–203
6. Rinderle, S., Reichert, M., Dadam, P.: Flexible support of team processes by adaptive workflow systems. Distributed and Parallel Databases **16** (2004) 91–116
7. Rinderle, S., Reichert, M., Dadam, P.: On dealing with structural conflicts between process type and instance changes. In: Proc. BPM'04. (2004) 274–289
8. Rinderle, S., Reichert, M., Dadam, P.: Disjoint and overlapping process changes: Challenges, solutions, applications. In: Proc. Int'l Conf. on Cooperative Information Systems (CoopIS'04). LNCS 3290, Larnaca, Cyprus (2004) 101–120
9. Weber, B., Wild, W., Breu, R.: CBRFlow: Enabling adaptive workflow management through conversational case-based reasoning. In: Proc. European Conf. on Cased based Reasoning (ECCBR'04), Madrid (2004) 434–448
10. Aha, D.W., Breslow, L., Muñoz-Avila, H.: Conversational case-based reasoning. Applied Intelligence **14** (2001) 9–32
11. Reichert, M.: Dynamic Changes in Workflow-Management-Systems. PhD thesis, University of Ulm, Computer Science Faculty (2000) (in German).
12. Rinderle, S.: Schema Evolution in Process Management Systems. PhD thesis, University of Ulm, Computer Science Faculty (2004)
13. Kolodner, J.L.: Case-Based Reasoning. Morgan Kaufmann (1993)
14. A. Aamodt, E.P.: Case-based reasoning: Foundational issues, methodological variations and system approaches. AI Communications **7** (1994) 39–59
15. Aha, D.W., Muñoz-Avila, H.: Introduction: Interactive case-based reasoning. Applied Intelligence **14** (2001) 7–8
16. Cheetham, W., Price, J.: Measures of solution accuracy in case-based reasoning systems. In: Proc. European Conf. on Case-Based Reasoning (ECCBR'04). LNCS 3155, Madrid (2004) 106–118

17. eBAY: Feedback Forum. (2005) http://pages.ebay.com/services/forum/feedback.html.
18. Kim, J., Suh, W., Lee, H.: Document-based workflow modeling: a case-based reasoning approach. Expert Systems with Applications **23** (2002) 77–93
19. Madhusudan, T., Zhao, J.: A case-based framework for workflow model management. In: Proc. 1st Int'l Conf. on Business Process Management (BPM'03), Eindhoven (2003) 354–369
20. Wargitsch, C.: Ein Beitrag zur Integration von Workflow- und Wissensmanagement unter besonderer Berücksichtigung komplexer Geschäftsprozesse. PhD thesis, Erlangen (1998)
21. Luo, Z., Sheth, A., amd J. Miller, K.K.: Exception handling in workflow systems. Applied Intelligence **13** (2000) 125–147
22. Limthanmaphon, B., Zhang, Y.: Web service composition with case-based reasoning. In: Proc. of 15th Australasian Database Conf. (ADC'02), Australia (2002)
23. Muñoz-Avila, H., McFarlane, D., Aha, D., Ballas, J., Breslow, L., Nau, D.: Using guidelines to constrain interactive case-based htn planning. In: Proceedings of the Third International Conference on Case-Based Reasoning, Munich (1999) 288–302
24. Veloso, M., Mulvehill, A., Cox, M.: Rationale-supported mixed-initiative case-based planning. In: Proceedings of the Ninth conference on Innovative Applications of Artificial Intelligence, Providence, Rhode Island (1997) 1072–1077
25. Muñoz-Avila, H., Aha, D., Nau, D., Breslow, L., Weber, R., Yamal, F.: Sin: Integrating case-based reasoning with task decomposition. In: Proc. IJCAI-2001, Seattle (2001) 99–104
26. Muñoz-Avila, H., Gupta, K., Aha, D., Nau, D.: Knowledge Based Project Planning. In: Knowledge Management and Organizational Memories. Kluwer Academic Publishers (2002)
27. Abecker, A., et al.: Enabling workflow-embedded OM access with the DECOR toolkit. In Dieng-Kuntz, R., Matta, N., eds.: Knowledge Management and Organizational Memories. Kluwer Academic Publishers (2002)
28. Elst, L., Aschoff, F., Bernardi, A., Maus, H., Schwarz, S.: Weakly-structured workflows for knowledge-intensive tasks: An experimental evaluation. In: Proc. 12th Int'l Workshop on Enabling Technologies. (2003) 340–345
29. Abecker, A., Bernardi, A., Hinkelmann, K., O. Kühn, O., Sintek, M.: Context-aware, proactive delivery of task-specific knowledge: The KnowMore project. Int. Journal on Information Systems Frontiers **2** (2000) 139–162