Towards Case-Based Adaptation of Workflows

Mirjam Minor, Ralph Bergmann, Sebastian Görg, Kirstin Walter

Business Information Systems II
University of Trier
54286 Trier, Germany
{minor|bergmann|goer4105|walt4701}@uni-trier.de

Abstract. Creation and adaptation of workflows is a difficult and costly task that is currently performed by human workflow modeling experts. Our paper describes a new approach for the automatic adaptation of workflows, which makes use of a case base of former workflow adaptations. We propose a general framework for case-based adaptation of workflows and then focus on novel methods to represent and reuse previous adaptation episodes for workflows. An empirical evaluation demonstrates the feasibility of the approach and provides valuable insights for future research.

1 Introduction

Today, many companies and organizations must be able to quickly adapt their business according to newly arising opportunities and demands from the customer. This requires a means to flexibly adapt the current business processes which guide the core activities of the organization. Workflow technology is widely used to control the execution of business processes. *Workflows* are “the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules” [1]. The basic constituents of workflows are *tasks* that describe an activity to be performed by an automated service (e.g. within a service-oriented architecture) or a human (e.g. an employee). The procedural rules for the tasks are usually described by routing constructs like sequences, loops, parallel and alternative branches that form a control flow for the tasks. A significant limitation of traditional workflow technology is its missing flexibility of the workflow concept and its management by workflow engines. Workflows are meant to be described in a modeling phase during the setup of the workflow system. Once described, they are executed repeatedly in the same manner over a long period of time. In order to address today’s requirements concerning easy and fast adaptation of workflows, a new class of workflow systems is currently emerging, called *agile workflows* systems [2, 3]. Agile workflow systems break the traditional separation between workflow modeling and execution. Workflows can be created (for example based on a template) for a particular demand or business case. Workflows can also be adapted even after they have already been started, for example if some unforeseen events occur. Hence, the creation and adaptation of workflows has now become a very important activity for which urgent support is needed.

Recent work on case-based reasoning (CBR) addresses these needs. Workflow retrieval [4, 5, 6, 7, 8] provides assistance to a workflow modeling expert. Instead of composing new workflows from scratch, retrieved workflows or portions of workflows can be reused. This already reduces the modeling effort significantly. However, the adaptation of the retrieved workflows during modeling or the adaptation of already started workflows is still a challenging task for workflow modelers. It is also an important and widely unexplored research topic for CBR.

This paper presents on a novel framework for the adaptation of workflows. The adaptation of a workflow itself is performed in case-based manner, which enables the reuse of adaptation experience. We collect the experience from previous workflow adaptation episodes in a dedicated adaptation case base. An adaptation case stores the aim of the performed adaptation (change request description), the original workflow, the adapted workflow as well as the difference of both workflows in terms of add and delete lists containing workflow elements. When a new adaptation request for a workflow occurs (as part of an initial workflow modeling activity or as consequence of an unforeseen event to a workflow that is already under execution) this adaptation is performed in a case-based way, i.e., by reuse of an adaptation case.

The next section of the paper discusses relevant related work. Then, we introduce the representation of adaptation cases and present in Sect. 4 the proposed CBR cycle for case-based adaptation. Sect. 5 describes briefly the retrieval of adaptation cases, while Sect. 6 provides the details of the workflow adaptation by adaptation case reuse. Finally, Sect. 7 describes the current state of the implementation of our approach as well as the results of an empirical study.

2 Related Work

Workflow-oriented CBR deals with CBR methods for cases representing workflows. The reuse of workflow templates is widely spread in recent commercial workflow management systems. Before a workflow is enacted, a new workflow instance is derived from the workflow template. CBR is a means to go beyond this kind of assistance. Case-based retrieval is employed to dynamically select and execute suitable workflows to support collaboration in business [11]. Montani [21] proposes the use of CBR to support the management of exceptions in business process execution. Within the WINGS workflow system a method for the semantic retrieval of data-centric workflows from a repository has been developed [7]. Conversational CBR has been applied in the tool CBRFlow [4] to guide the user in adapting a workflow to changing circumstances. Leake and Morwick [8] evaluate the execution paths of past workflows in order to support user extension of workflows that are under construction. Several approaches exploit the relationships between plan and workflow representations and apply case-based planning (CBP) methods (see [9, 10] for a survey on CBP) for the construction of workflows. The CODAW system [5] supports the incremental modeling of workflows by similarity-based retrieval and workflow composition using an HTN planner. Xu and Munoz-Avila [22] apply case-based HTN planning for the construction of project plans. Most approaches of this kind rely on a first principles planner as underlying methodology being enhanced by an adaptation approach. Thus, the strong requirement of a formal planning model (e.g. some kind of STRIPS-like formalization) holds as well for the workflow steps.

This paper focuses on adaptation in the context of workflow cases. Several techniques for adaptation in CBR have been proposed so far (for a review see [12]). The most basic distinction between different adaptation methods is whether transformational adaptation or generative adaptation is applied. Transformational adaptation adapts the solution directly based on the differences between the current problem and the problem stated in the case. This is a knowledge intensive approach since domain specific adaptation knowledge is required that describes how differences in the problem should be compensated by modifications of the solution. Various methods have been proposed which differ in the representation and processing of such adaptation knowledge. Generative adaptation methods require a generative problem solver (e.g. an AI planner) that is able to solve problems based on a complete and sound domain model. This is a very strong assumption that inhibits the application of this approach in many typical CBR application domains where no such domain model can be built. More recent approaches to adaptation in CBR are motivated by the fact that the acquisition of explicit adaptation knowledge for transformational adaptation is a very difficult and time consuming task. Several methods have been developed that exploit the knowledge already captured in the cases as source of adaptation knowledge. In our own earlier previous work [13], we propose a general framework for learning adaptation knowledge from cases. Meanwhile, several successful examples of such methods exist that either apply inductive learning to extract adaptation knowledge [14, 15, 16] or that apply a case-based adaptation approach [17].

The adaptation approach proposed in this paper is a kind of case-based adaptation as it is based on a dedicated case base of previous successful adaptations. On the other hand, each adaptation case describes a particular kind of workflow modification that can be transferred to new situations. Hence, the adaptation case base can be considered experiential transformational adaptation knowledge. In particular, we do not assume any formal semantic representation of the workflow steps (tasks).

3 Representation of Workflow Adaptation Cases

The adaptation knowledge for workflows is represented in case-based manner. Before the case structure is discussed, the workflow modeling language we have developed within the CAKE framework [3,11] is briefly introduced. However, our adaptation approach can be extended to other control-flow-oriented workflow languages.

The CAKE workflow modeling language (compare [3]) consists of several types of *workflow elements* whose instances form block-oriented control flow structures. A workflow element can be a task, a start element, an end element, or a control flow element like an AND, XOR, loop etc. Control flow elements always define related blocks (AND-block, XOR-block etc.). Blocks cannot be interleaved but they can be nested. For example, the workflow depicted in box 2 of Fig. 1 has the following workflow elements: A start element, a task “Ask for invoice number”, an AND-block containing two further tasks, a task “Send confirmation”, and an end element. Each workflow element has a unique identifier (ID) and an execution state. The execution state “ACTIVE” is assigned to elements that are currently executed. “READY” means that a workflow element has not yet been executed, and “COMPLETED” denotes the status of workflow elements that have already finished execution.

An *adaptation case* can now be described as follows. It is the representation of a previous adaptation episode of a workflow. Like a traditional case, it is structured into a problem part and a solution part:

1. The *problem part* consists of
   a. a semantic description of the *change request* (what causes the change) and
   b. the *original workflow* prior to the adaptation.
2. The *solution part* contains
   a. the *adapted workflow* and
   b. the description of the *adaptation steps* (added and deleted workflow elements) that have been executed to transform the original workflow into the adapted workflow.

The change request is a semantic description of the cause of the request. This part of the problem description makes use of traditional case representation approaches, e.g. a structural representation by meta data or a textual representation. Workflows are represented as described before. The representation of the adaptation steps (2b)

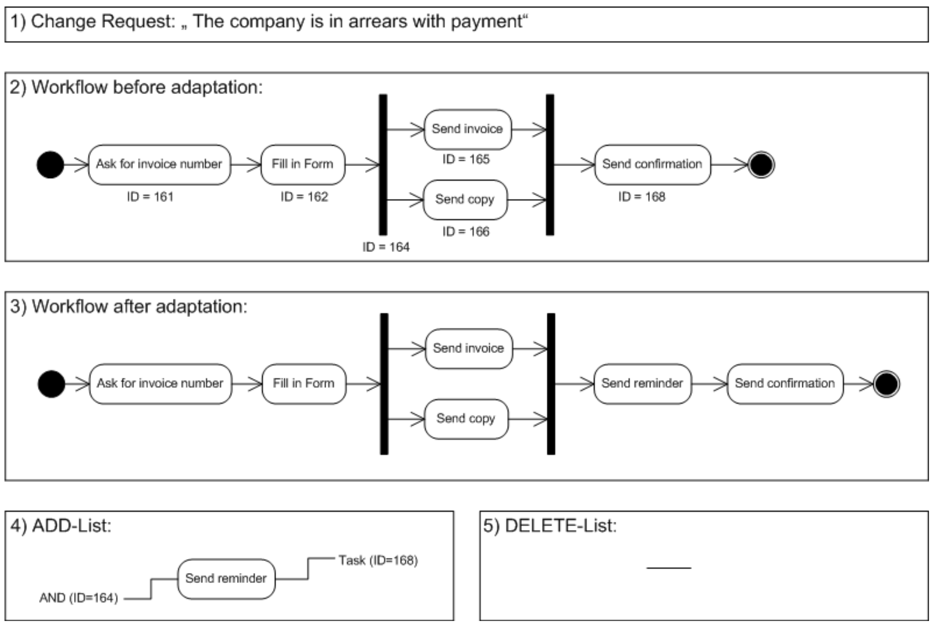

Fig. 1 Sample workflow adaptation case for the workflow "Send an invoice".

deserves some special attention. Similar to STRIPS operators, the adaptation steps are described by an *add* and a *delete* list. Each list contains a set of chains of workflow elements. A chain encapsulates a maximum set of connected workflow elements with one entry point and one exit point. Further, each chain records a pair of *anchors*. A *pre anchor* is the workflow element (in the original workflow) after which workflow elements from the chain have been added or deleted. A *post anchor* is the workflow element from the original workflow following the last element of the chain. Hence, the pre anchor describes the position after which the adaptation starts and the post anchor describes the first position in the original workflow that is not anymore affected by the adaptation described in the chain. These anchors are further used during the reuse of the workflow adaptation to identify similar points in new workflows at which the proposed adaptation can be applied. Please note that the overall description of the adaptation steps can be automatically computed from the original workflow and the adapted workflow or alternatively it can be captured as part of a modeling tool that is used for manual workflow adaptation.

Fig. 1 illustrates a sample workflow adaptation case. The considered workflow describes the business process of invoicing something to a business partner. The topmost box of Fig. 1 contains a sample change request (in a textual representation) that occurred during the execution of the invoice workflow: The recipient of the invoice is in arrears with the payment. The original workflow is depicted in box 2 of Fig. 1 in UML activity diagram notation. This workflow is adapted to fulfill the change request, by sending a reminder to the partner. The workflow in box 3 shows the adapted workflow. The original workflow has been modified by adding the newly created task “Send reminder”. In boxes 4 and 5 the according adaptation steps are represented by an add and a delete list. The added task “Send reminder” is listed together with the pre anchor (AND with element ID 164) and the post anchor (Send confirmation Task with element ID 168).

4 CBR Cycle as Framework for Case-Based Adaptation

We now propose a general framework for the case-based adaptation of workflows by applying the traditional CBR cycle [18] to adaptation cases. Hence, the case base consists of adaptation cases (see Sect. 5). The similarity measure, which must be able to assess the similarity between two problem descriptions of an adaptation case must therefore be able to consider the *change request similarity* as well as the *workflow similarity*.

The new problem to be solved (query) consists of a *target workflow* (which may be already partially executed) and the description of the *current change request* of the change to be made on this target workflow. Fig. 2 illustrates the framework. In the first step the most relevant adaptation cases are *retrieved* from the case base by similarity-based retrieval employing the similarity measure. One or possibly several adaptation cases are selected for reuse. In the *reuse* phase, the adaptation steps contained in the adaptation case(s) are applied to the target workflow. This occurs in two distinct steps. First, the concrete location in the target workflow is determined that needs to be changed. This is necessary as there are usually many different positions within the workflow at which a chain from the add list can be inserted or to which the deletions in the delete list can be applied. Second, the changes are applied to the target workflow at the determined locations. The resulting *adapted workflow* is then the proposed solution. During the subsequent *revise* phase the user can validate the workflow adaptations proposed: she can either confirm them or revise them by manually performing appropriate adaptations herself. The current graphical workflow modeling component of our CAKE system [19] supports such a manual workflow adaptation. The resulting confirmed solution then represents the correctly adapted target workflow that the workflow engine continues to enact. Finally, a new validated adaptation case has been constructed, which can then be *retained* as part of the adaptation case base.

We see that the application of the CBR cycle to adaptation cases is quite straight forward. However, the most crucial part is the reuse phase which requires novel methods for transforming previous workflow modifications in order to apply them to the target workflow. Therefore, our research in this paper focuses particularly on this phase. Only after the characteristics of the new reuse methods have been investigated, we will be able to re-consider the similarity assessment as the similarity assessment must ensure the retrieval of adaptable cases and hence it is dependent on the particular reuse methods. Despite of the focus on reuse methods we will now briefly give some thoughts on similarity measures for workflows (Sect. 5) as they are also needed within the reuse methods being described in Sect. 6.

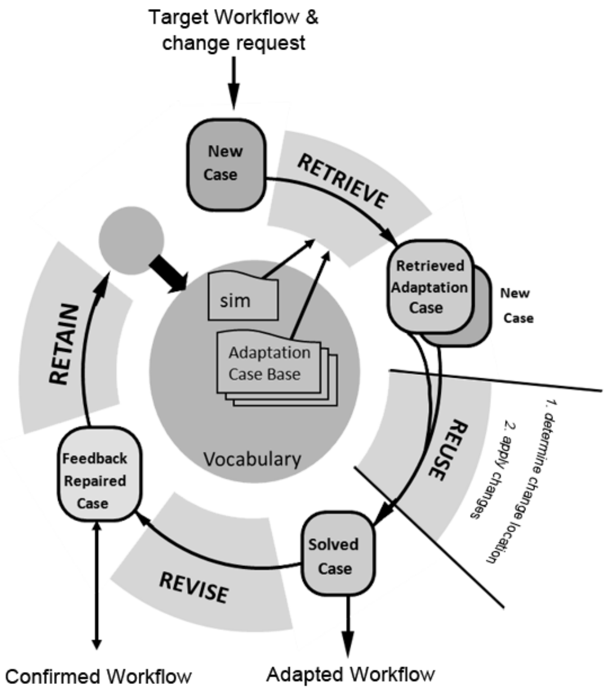

Fig. 2. Visualization of the adaptation cycle.

5 Workflow Retrieval

The retrieval of workflow adaptation cases requires similarity measures for the different components of the problem part: a *change request similarity* and a *workflow similarity*. Both similarity values are aggregated (we use a weighted sum) to determine the overall similarity of adaptation cases. As the focus of this paper is an adaptation, we only sketch a very simple similarity measure, which we use in the subsequent case study. Please refer to our previous work [3] and to the literature [5] for more sophisticated similarity measures for workflows that could be used as well. For the purpose of his work, we restrict our representation of the change request to a textual description of the purpose of the change. Change request similarity is then derived from the traditional Levenshtein distance measure. The Levenshtein distance is purely syntactic and measures the minimum number of edit operations on the character level that is required to transform one string into another. Future work will elaborate the use of a change request ontology (an a related similarity measure) to include more semantics.

The workflow similarity measure used aggregates the similarity of the set of occurring workflow elements and the similarity of the abstracted control flow structure. The similarity measure for workflow elements (which is also employed during the adaptation, see Sect. 6) compares tasks by means of their task names and control-flow elements (AND, XOR, etc.) by an exact matching. The similarity of task names is again simply computed by a Levensthein distance. In future work, more sophisticated measures for the similarity of workflow elements could be investigated. For instance, further properties of the workflow elements could be included like the

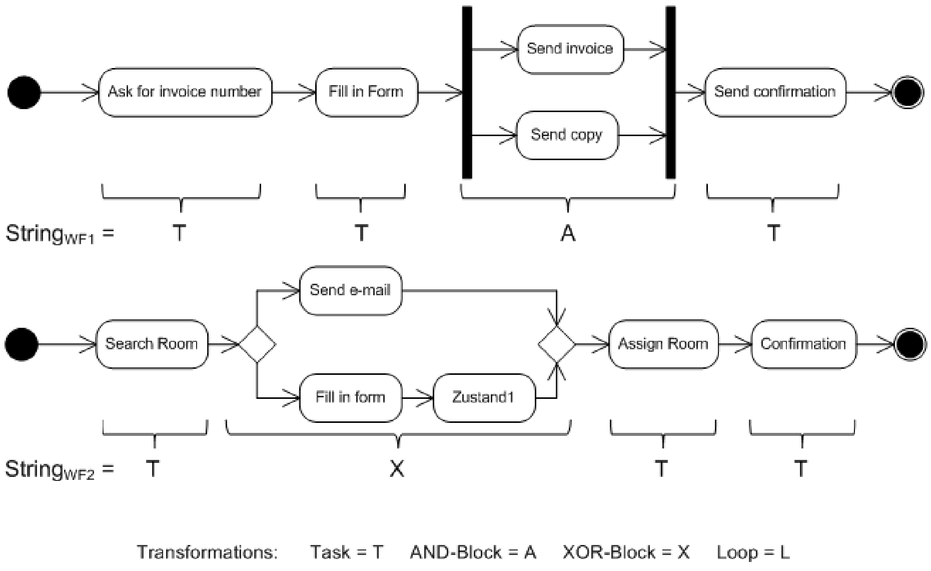

Fig. 3. Simple serialization of the control flow.

input and output parameters specifying the data flow.

In order to compute the similarity of the abstracted control flow the control flow graph is serialized into a string containing a specific character for each workflow element (see Fig. 3 for an example). For reasons of simplicity, the current serialization only considers the utmost sequence of tasks and blocks, but can be easily extended to include inner blocks as well. The similarity between two serialization strings is again measured by applying the Levenshtein distance. For the sample in Fig. 3, the Levenshtein distance is 2, which would induce a similarity value of 0.5.

6 Workflow Adaptation

The result of the workflow retrieval is one or more workflow adaptation cases. In the current approach, we accept only one adaptation case per reuse phase. The adaptation knowledge from this case is to be transformed and applied to the new workflow. The assumption is that the lists of edit operations from the retrieved case (add and delete list) will provide appropriate structural changes when adopted to modify the new workflow. Two main issues have to be solved for this: First to determine the locations where the adaptation steps should be applied to the new workflow and, second, to execute the adaptation.

6.1 Determine Change Location

In a manual adaptation scenario the workflow modeler knows where a workflow needs to be adapted. For the scenario of an automatic workflow adaptation, the positions for applying the edit operations in the target workflow have to be determined for every chain. This problem can be solved by means of the anchor concept. The anchors of the chains have to be mapped to appropriate positions in the target workflow.

The anchor mapping method consists of three steps: (1) First it determines *candidate positions* for the anchors in the target workflow. (2) Then it assesses the candidate positions to restrict the set of potential positions to a set of *valid candidate positions* for each anchor. (3) Last, the mapping from the anchors to the set of valid candidate positions is constructed. The resulting mapping might be partial but it has to satisfy the following consistency criterion: A mapping of anchors to positions in the target workflow is called *consistent* if it preserves the inner order of anchor pairs. That means that for any mapped anchor pair the position of the pre anchor must be before to the position of the post anchor in the target workflow. A pair of candidate positions is called *consistent pair of candidate positions* if it does not violate the consistency of a mapping. Fig. 4 illustrates the mapping method by the sample anchor pair from Fig. 1. The target workflow provides six candidate positions. The result of the mapping method is only a partial mapping as the pre anchor could not be mapped to an appropriate position. In the following, the three steps of the mapping method will be described in more detail.

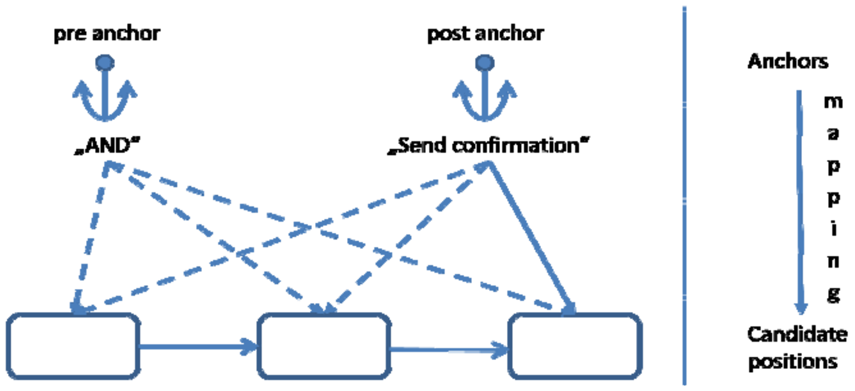

Fig. 4. Sample candidate positions for a pair of anchors.

(1) The first task of determining candidate positions uses a filter for workflow elements that provides candidate positions. It would not be useful to take workflow elements into consideration that have already completed their execution. Hence, the execution state of the workflow elements is evaluated to filter out workflow elements that are not editable any more (those with the status ACTIVE or READY remain). The remaining workflow elements provide the candidate positions. This first step of the mapping method can be omitted in case the workflows have not yet started execution. In this case, the positions of all workflow elements in the target workflow are regarded candidate positions.

(2) The second step of the mapping method assesses the candidate positions for a particular anchor. The assessment step employs the similarity measure for workflow elements to filter out invalid candidate positions. The similarity of the element at the candidate position to the element at the anchor position in the chain is computed. All candidate positions for which this similarity value is above a threshold called the *validity threshold* are considered valid candidate positions for an anchor. The other candidate positions are refused.

(3) The third step selects valid candidate positions to construct a consistent mapping of anchors to candidate positions. Four different mapping methods have been investigated (compare Sect. 7):
   - **Pre anchor mapping (PRAM)**: The post anchors are ignored, while for each pre anchor the best matching candidate position is selected according to the workflow similarity measure. In case a candidate position is the best matching position for multiple pre anchors, one pre anchor is chosen arbitrarily and the others are mapped to the next best matching candidates.
   - **Post anchor mapping (POAM)**: The pre anchors are ignored, while the best matching positions for the post anchors are selected analogously to the PRAM method.
   - **Composite anchor mapping (CAM)**: The mapping chooses candidate positions from the set of consistent pairs of valid candidate positions and

from the set of single candidates as follows. First the set of pairs is split into two sets, namely to the set of *tight pairs*, that means that the candidate position of the pre anchor is a direct successor to the candidate position of the post anchor, and to the set of non-tight pairs. The set of tight pairs provides candidates for mapping the anchors from the add list, while the set of non-tight pairs is considered for anchors from the delete list. Then both sets of pairs are ordered according to the sum of the workflow element similarities at the pre anchor positions and at the post anchor positions. The similarity value of the best matching pair (in case there is one) is compared to the similarity value of the best matching single candidate according to the PRAM and PROM method. The pair or single candidate with the highest similarity value is chosen. In case a candidate is selected several times, one anchor pair is selected arbitrarily and the others are mapped to the next best matching pairs or single candidates.

- **Maximum anchor mapping (MAM)**: This method is very similar to the CAM method. Just the ordering of candidate pairs is different: The sum aggregating the similarity values for the pre and post anchor positions of a pair is replaced by a maximum function.

The mapping result plays a different role for chains from add lists and from delete lists: In case of a chain from the add list, the anchor positions are the only criterion to determine the insert positions within the target workflow. In case of a chain from the delete list, the main criterion to determine the positions of workflow elements that have to be deleted from the new workflow is a mapping of the workflow elements from the delete list to workflow elements in the target workflow. Again, the similarity measure for workflow elements is applied. The best matching workflow elements are candidates for being deleted. The fitting anchor positions are a secondary criterion to prevent premature delete operations.

6.2 Application of Changes

The positions of the anchors determined by the mapping method specify where to apply the changes to the target workflow. The add operations are executed immediately for all chains of which at least one anchor has been mapped successfully. In case there is a position for the pre anchor the add operations are performed starting at this point. Otherwise, the add operations are performed ending at the position of the post anchor. Chains whose anchors could not be mapped cannot be applied automatically. They might be applied by the user in the revise phase (see Sect. 4).

Chains from the delete list are applied only if the mapping of their elements is complete and the similarity values for the individual workflow elements are above a threshold called *delete threshold*. Additionally, it is required that the elements to be deleted are organized in exactly the same control flow than those in the chain. Thus, the delete operations have to fulfill stronger constraints than the add operations to be applied.

The above list of four mapping methods may be extended by further methods. None of the above methods guarantees completeness but partial mappings allow to apply parts of the adaptation steps. As the evaluation has shown (see below), the application of the four methods has been quite successful already for some test cases.

7 Implementation and Evaluation

7.1 Collaborative Agent-based Knowledge Engine (CAKE)

The described methods have been fully implemented as part of the Collaborative Agent-based Knowledge Engine (CAKE) [11, 19] developed at the University of Trier within several research projects. CAKE provides modeling and enactment support for agile workflows. It uses a light-weight process modeling language for the agile workflows at the user level by extending the UML activity diagram notation by own symbols (see also Sect. 3). The workflow technology offered allows late-modeling and manual adaptation of ongoing workflows. Some typical ad-hoc changes are to re-order some parts of a workflow instance or to insert an additional task. CAKE also offers data modeling facilities using a common object-oriented data model and a comprehensive structural case representation framework, a library of configurable similarity measures, and a similarity-based retrieval component. As part of our current research CAKE has been extended as follows:

- it now supports representation of adaptation cases
- the application of the existing similarity measures for the retrieval of adaptation cases, and
- the reuse of adaptation cases as described in Sect. 6.

7.2 Evaluation Goal and Domain

The extended CAKE system has then been used in a first empirical evaluation of the proposed adaptation methods. The evaluation itself is a big challenge, since immediately useful test data about workflow cases is not available. Although some public workflow repositories exist, none of them considers and records workflow changes. Therefore, we need to spend a significant effort to develop and elaborate a particular test scenario for this evaluation. The evaluate our approach, it was necessary to obtain a case base of adaptation cases and a set of new test problems each of which consists of a target workflow and a change request. We decided to perform this evaluation in the domain of administrative processes as they typically occur in University offices. In several structured interview sessions with office secretaries, we first obtained a set of 19 typical office workflows as they are executed on a regular basis in our University. Then, for each workflow the interviewees were asked to describe a common change request and the resulting change in the workflow [19]. From the records of these interviews, a case base of adaptation cases has been manually constructed using the CAKE system. Then, in an independent interview session, a set of nine new test problems together with reference solutions (adapted workflows) was also elicited from the secretaries. These test queries have been obtained in a similar manner than the adaptation cases, but care has been taken that the workflows are different from those in the case base. We are aware that the size of the case base and the number of target problems is quite small and does not allow applying statistical methods to evaluate the results, but the large effort of such an experiment with real users does not yet enable a larger experimental data set in this phase of research. However, a initial evaluation and an assessment of the applicability of this approach for the domain of office administration should be possible.

As the target of our research is the reuse phase, this evaluation particularly aims at providing first insights about the four variants of workflow adaptation. We want to investigate whether the four methods are able to correctly apply the adaptation steps in the adaptation cases and whether the four methods differ in the degree of correctness of the adaptation. For this purpose we executed two experiments described below.

7.3 Experiment 1: Adaptation Case Reconstruction

The aim of the first experiment was to evaluate whether the each of the proposed reuse methods is able to correctly reconstruct the adaptations described in each of the 19 adaptation cases. For each adaptation case the four variants of reuse methods have been applied to the contained original workflow, i.e. the description of the adaptation steps is used to compute an adapted workflow from the original workflow. By comparing the computed adapted workflow with the correct adapted workflow noted in the adaptation case, the correctness of the reuse method can be measured. The correctness then mainly depends on the capability to correctly reconstruct the anchor positions to which the adaptations have to be applied. As a quality criterion we measured the average number of *correctly* applied *add* chains, *wrongly* applied add chains, and *not* applied add chains as well as the average number of *correctly* applied *delete* chains, *wrongly* applied delete chains, and *not* applied delete chains. Fig. 5 shows the results of this experiment. There are two bars for each variant. The left bar shows the results for the 22 add chains and the right bar shows the results for the 7

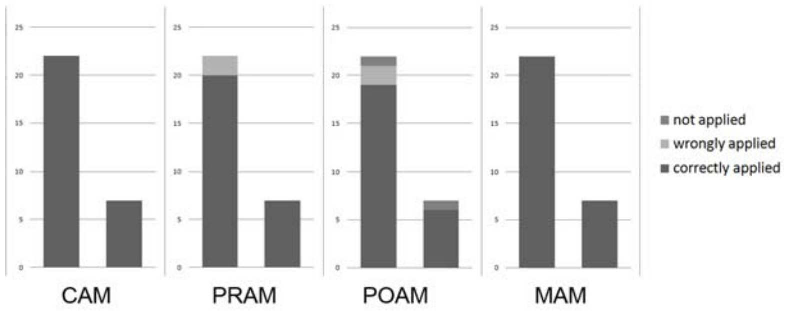

Fig. 5. Results of Experiment 1.

delete chains occurring altogether in the 19 adaptation cases. Only CAM and MAM are able to correctly reconstruct all adaptation cases. If only one anchor is used as in PRAM and POAM, not all anchor positions can be identified correctly.

7.4 Experiment 2: Adaptation of Test Problems

In the second experiment, we investigated the capability of the four reuse variants to correctly adapt the workflows of the 9 test problems. As we only aim at evaluating the reuse phase, we skip the retrieval phase of our adapted CBR cycle from Fig. 2 to diminish the impact of an improper similarity measure on the assessment of the reuse methods. Instead, for each test problem, the best matching adaptation case has been selected manually with support from an office secretary. Then, the adaptation steps in the selected case are applied to the target workflow of the test problem. In this experiment, we used a high validity threshold of 0.9 for add lists. Hence, adaptations are only performed if there is high evidence that the positions are correct. Finally, the resulting adapted workflow is compared with the reference solution noted within the test problem description. Then, the same quality criterion is applied as in experiment 1. Fig. 6 illustrates the results of this experiment. As in experiment 1 the left bar shows the results for the add-chains and the right bar the results for the delete-chains. The best results are achieved by CAM and MAM. However, there is no difference of the four variants applied to the delete chains and there is only little differences of CAM and MAM applied to the add chains.

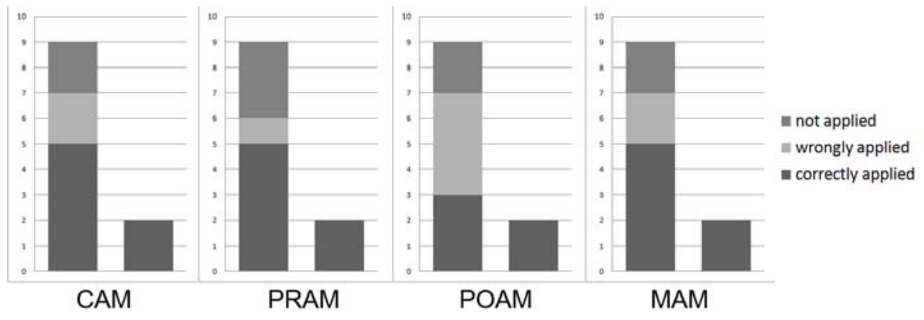

Fig. 6. Results of Experiment 2 - validity threshold=0.9.

To investigate the influence of the validity threshold, we lowered the validity threshold to 0.7 and run the experiment again. With a lower threshold, adaptation steps are transferred even if the matching of the anchors is less perfect, leading to increased matching opportunities. If the similarity measure is appropriate, we expect also an increasing number of correctly applied adaptation steps.

The results are shown in Fig. 7. The best result was obtained by CAM leading to 7 correctly applied and none not applied add list. With all four methods, the number of correctly applied adaptation steps is increased with the lower threshold, while the number of wrongly applied steps is not changed. These results indicate that at least in this setting a lower validity threshold improves finding proper anchors. It is quite

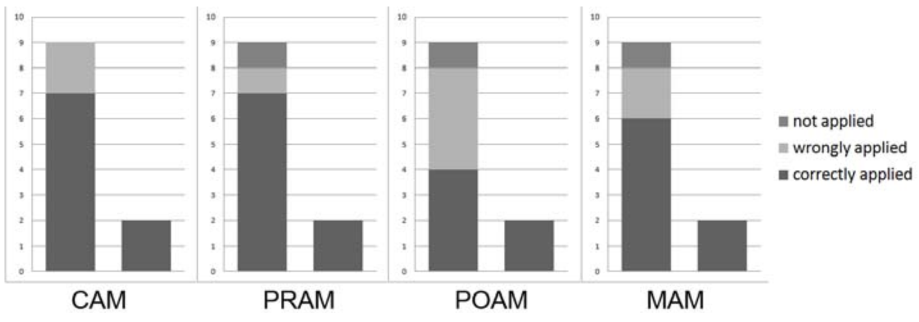

Fig. 7. Results of Experiment 2 - validity threshold=0.7

likely that this effect is strongly dependent on the similarity measure used. This issue must be investigated in more depth in future experiments with improved similarity measures.

8 Conclusion and Open Issues

In this paper, we presented a novel framework for case-based adaptation of workflows together with first results from an empirical evaluation of a sample implementation. Workflow adaptation cases store pairs of workflows and a change request describing the cause why the first workflow has been converted into the second. The adaptation steps are described as edit operations (add and delete steps) that are applicable to similar workflows with similar change requests as well. The reuse phase consists of two tasks: First to determine the change location in the target workflow and second to apply the changes from the retrieved case. A novel anchor concept has been introduced to find appropriate locations for chains of edit operations. The evaluation on sample workflows from the office domain has shown that considering two anchors per chain performs better results than single anchors only. The anchor concept has turned out to be suitable for transforming adaptation steps from one workflow to another in a real-world domain. Our work so far has confirmed the intention to develop a framework for case-based adaptation of workflows. Novel open issues arise that concern the reuse phase (e.g. more complicated anchors integrating several workflow elements and data flow) as well as general issues like the acquisition and maintenance of the adaptation cases. Further domains like e-science workflows or cooking could be investigated. We believe that our results could be useful for the procedural knowledge in many further application areas.

9 References

1. Workflow Management Coalition: Workflow management coalition glossary & terminology, http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf, 2007.
2. Weber, B., Wild, W.: Towards the agile management of business processes. *WM'05, Revised Selected Papers*, LNCS 3782, pp. 409-419, Springer, 2005.

3. Minor, M., Tartakovski, A., Schmalen, D., Bergmann, R.: *Agile Workflow Technology and Case-Based Change Reuse for Long-Term Processes*. International Journal on Intelligent Information Technologies 4(1), pp. 80-98, 2008.
4. Weber, B., Wild, W., Breu, R.: Cbrflow: Enabling adaptive workflow management through conversational case-based reasoning. *ECCBR'04,* LNAI 3155, pp. 434-448, Springer 2004.
5. Madhusudan, T., Zhao, J.L., Marshall, B.: *A case-based reasoning framework for workflow model management.* Data and Knowledge Engineering, 50: pp. 87-115, 2004.
6. Minor, M., Tartakovski, A., Bergmann, R.: Representation and Structure-Based Similarity Assessment for Agile Workflows. *ICCBR'07,* pp. 224-238, LNCS 4626, Springer, 2007.
7. Gil, Y., Kim, J., Florenz, G., Ratnakar, V., Gonzalez-Calero, P.: Workflow Matching Using Semantic Metadata. *Proc. of the 5th Int. Conf. on Knowledge Capture (K-CAP),* 2009.
8. Leake, D.B., Kendall-Morwick, J.: Towards Case-Based Support for e-Science Workflow Generation by Mining Provenance. *ECCBR'08,* LNCS 5239, pp. 269-283, Springer 2008.
9. Cox, M.T., Muñoz-Avila, H., Bergmann, R.: *Case-based planning*. The Knowledge Engineering Review, 20(3), pp. 283-287, 2005.
10. Muñoz-Avila, H., Cox, M.T.: *Case-based plan adaptation: An analysis and review*. IEEE Intelligent Systems, 23(4), pp. 75-81, 2008.
11. Bergmann, R., Freßmann, A., Maximini, K., Maximini, R., Sauer, T.: Case-based support for collaborative business. *ECCBR'06,* LNAI 4106, pp. 519-533. Springer, 2006.
12. Lopez Mantaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M.T., Forbus, K., Keane, M., Aamodt, A., Watson, I.: *Retrieval, reuse, revision and retention in case-based reasoning*. The Knowledge Engineering Review, 20(03), pp. 215-240, 2005.
13. Wilke, W., Vollrath, I., Bergmann, R.: Using knowledge containers to model a framework for learning adaptation knowledge. *ECML Workshop Notes*. LIS, Faculty of Informatics and Statistics, Prague, 1998.
14. Hanney, K., Keane, M.: The adaptation knowledge bottleneck: How to ease it by learning from cases. *ICCBR'97*, LNAI 1266, pp. 179-192, Springer, 1997.
15. Craw, S., Wiratunga, N., Rowe, R.C.: Learning adaptation knowledge to improve case-based reasoning. *Artificial Intelligence,* 170(16-17), pp. 1175-1192, 2006.
16. Badra, F., Cordier, A., Lieber, J.: Opportunistic adaptation knowledge discovery. *ICCBR'09,* LNCS 5650, pp. 60–74, Springer, 2009.
17. McSherry, D.: Demand-driven discovery of adaptation knowledge. *IJCAI'99,* pp. 222-227. Morgan Kaufmann, 1999.
18. Aamodt, A., Plaza, E.: *Case-based reasoning: Foundational issues, methodological variations, and system approaches*. AI Communications, 7(1), pp. 39-59, 1994.
19. Minor, M., Schmalen, D., Kempin, S.: Demonstration of the Agile Workflow Management System Cake II Based on Long-Term Office Workflows. *CEUR proceedings of the BPM'09 Demonstration Track*, volume 489, 2009.
20. Minor, M., Schmalen, D., Weidlich, J., Koldehoff, A.: Introspection into an agile workflow engine for long-term processes. *WETICE'08,* IEEE Computer Society, Los Alamitos, 2008.
21. Montani, S.: *Prototype-based management of business process exception cases*. Applied Intelligence. Springer, 2009.
22. Xu, K., Munoz-Avila, H. CaBMA: Case-Based Project Management Assistant. *Proceedings of The Sixteenth Innovative Applications of Artificial Intelligence Conference* (IAAI-04). AAAI Press, 2004.