Similarity-Based Retrieval and Automatic Adaptation of Semantic Workflows

Ralph Bergmann and Gilbert Müller

Abstract The increasing demand for individual and more flexible process models and workflows asks for new intelligent process-oriented information systems. Such systems should, among other things, support domain experts in the creation and adaptation of process models or workflows. For this purpose, repositories of best practice workflows are an important means as they collect valuable experiential knowledge that can be reused in various ways. In this chapter we present process-oriented case-based reasoning (POCBR) as a method to support the creation and adaptation of workflows based on such knowledge. We provide a general introduction to process-oriented case-based reasoning and present a concise view of the POCBR methods we developed during the past ten years. This includes graph-based representation of semantic workflows, semantic workflow similarity, similarity-based retrieval, and workflow adaptation based on automatically learned adaptation knowledge. Finally, we sketch several application domains such as traditional business processes, social workflows, and cooking workflows.

1 Introduction

Business process management is a well-established discipline that deals with the identification, modeling, analysis, improvement, and implementation of business processes [1]. It is a methodology that is widely applied today to improve the operation of organizations and to align the IT development with business processes. Workflow management is a specific area of business process management that aims at "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

R. Bergmann · G. Müller (✉)
Business Information Systems II, University of Trier, 54286 Trier, Germany
e-mail: muellerg@uni-trier.de

R. Bergmann
e-mail: bergmann@uni-trier.de

G.J. Nalepa and J. Baumeister (eds.), *Synergies Between Knowledge Engineering and Software Engineering*, Advances in Intelligent Systems and Computing 626, https://doi.org/10.1007/978-3-319-64161-4_2

to a set of procedural rule" [2]. In the recent years, the use of workflows has significantly expanded from the original domain of business processes towards new areas, which also manifests their relevance in software engineering and development in various ways. For example, in engineering domains, such as software development or chip design, workflows are used to support complex collaborative and distributed development processes [3, 4]. In e-science scientific workflows are executable descriptions of computable scientific processes (i.e. a kind of executable programs) such as computational science simulations and data analyses [5]. Furthermore, workflows can be used to represent and automatically execute search [6] and information integration processes [7] in the context of decision support systems. Even for everyday activities such as cooking, workflows can be used as a means to represent the cooking instructions within a recipe [8] in order to provide step-by-step guidance for the chef.

One of the biggest challenges today arises from the fact that many companies and organizations must be able to more quickly adapt their business according to newly arising market opportunities and demands from the customer, due to actions of the competitors, or due to new technological developments. Agility became an important requirement in may domains in which workflows are applied [1, 9, 10]. Thus, instead of using a small set of standardized workflows, there is an increasing demand for tailoring workflows in a case-specific manner according to the current needs. This asks for intelligent, knowledge-based systems that assist domain experts in the creation or adaptation of workflows. Such systems must be able to represent and reason with knowledge about workflows and workflows elements, such as task, and data items. They must include knowledge that allows to assess the utility of workflows with respect to certain user demands, and they must possess knowledge about appropriate ways to adapt workflows. Consequently, the development of such knowledge-based systems involves a significant knowledge engineering effort that asks for methods from knowledge acquisition, semantic technologies, and machine learning.

In this chapter we present process-oriented case-based reasoning (POCBR) as a method to support the creation and adaptation of workflows. Case-based reasoning (CBR) is an established Artificial Intelligence methodology for experience-based problem-solving by selecting previous problem solutions from the past and adapting them to address a current but related problem [11]. POCBR is a specific sub-branch of CBR that deals with knowledge about processes and workflows [12]. In our own research within the past 10 years, we developed several POCBR methods as well as a generic system called CAKE (Collaborative Agile Knowledge Engine) [13] that support retrieval and adaptation of workflows. Here, experiential knowledge is stored in a repository and consists of semantic workflows, which are best-practices workflows from the past that are semantically annotated using concepts from a domain ontology in order to support their reuse. Users can query the repository with a specification of important properties of the workflow s/he wants to create in order to retrieve potentially reusable workflows. Workflow adaptation methods can then be applied to automatically adapt the retrieved workflow towards the user's query.

In the next section we provide a general introduction into POCBR. Section 3 describes the technical foundations for semantic workflows. The similarity-based retrieval of reusable workflows from a repository is described in Sect. 4, while Sect. 5 presents three different methods for knowledge-based adaptation of workflows. An overview of the CAKE system and selected application examples are described in Sect. 6.

2 Process-Oriented Case-Based Reasoning

Case-based reasoning (CBR) is a problem solving paradigm built upon a rule of thumb suggesting that similar problems tend to have similar solutions [11, 14]. The core of every case-based reasoning system is a case base, which is a collection of memorized experience, called cases. The CBR cycle proposed by Aamodt and Plaza [11] consists of four CBR phases, performed sequentially when a new problem (also called new case or query) must be solved [15]. First, the *retrieve phase* selects one or several cases from the case base with the highest similarity to the query, where similarity is determined by an underlying similarity measure [14]. In the subsequent *reuse phase*, the solutions of the retrieved cases are adapted according to the requirements of the query. In the *revise phase*, the solution determined so far is verified and possibly corrected or improved, e.g., through intervention of a domain expert. Finally, the *retain phase* takes the feedback from the revise phase and updates the knowledge of the CBR problem solver. As part of this learning phase, cases can be added to or deleted from the case base, but also other kinds of knowledge, such as similarity measures or adaptation knowledge can be affected. A unified view on the knowledge contained in a CBR application was proposed by Richter [14] through the metaphor of the four *knowledge containers*: the vocabulary, the case base, the similarity measure, and the adaptation knowledge. The *vocabulary* (which is typically called ontology today) is the basis of all knowledge and experience representation in CBR. The vocabulary defines the information entities and structures (e.g., classes, relations, attributes, data types) that can be used to represent cases, similarity measures, and adaptation knowledge. The *case base* is the primary form of knowledge in CBR, i.e., a repository of cases. A *case* is the representation of a specific experience item (e.g. a problem-solution pair, a problem-solving trace, or a best-practice procedure for performing a certain job) using the predefined vocabulary. The notion of *similarity* plays a key role in CBR, since cases are selected based on their similarity to the current problem. While early CBR approaches were usually restricted to standard similarity measures (such as inverse Euclidean or Hamming distances), the current view is that the similarity measure encodes important knowledge of the domain. Several techniques for *adaptation* in CBR have been proposed so far [15]. However, all adaptation methods require appropriate additional knowledge for adaptation. Motivated by the fact that the manual acquisition of adaptation knowledge is very difficult, several methods have been developed that exploit the knowledge already captured in the cases as source to automatically learn adaptation knowledge

[16, 17]. As a specific approach to knowledge engineering and knowledge-based systems design, CBR is closely related to analogical reasoning, machine learning, information retrieval, databases, semantic web, and knowledge management.

Process-oriented CBR (POCBR) addresses the integration of CBR with process-oriented research areas like Business Process Management and Workflow Management (WFM) [12]. In POCBR a case is usually a workflow or process description expressing procedural experiential knowledge. POCBR aims at providing experience-based support for the automatic extraction [18], design [19], execution [20], monitoring and optimization [21, 22] of workflows. In particular, new workflows can be constructed by reuse of already available workflows that are adapted to new purposes and circumstances. Thereby, the laborious development of workflows from scratch can be avoided.

A case base (or repository) of successful workflows reflecting best-practices in a domain is the core of a POCBR approach. Users can query the repository with a specification of important properties of the workflow s/he wants to create in order to retrieve potentially reusable workflows. One particular characteristic of CBR is that it allows to find cases that do not match exactly the user's query, but which are at least similar in some respect. For example, the CODAW system [23] supports the incremental modeling of workflows by similarity-based reuse of the workflow templates. Leake and Morwick [19] evaluate the execution paths of past workflows in order to support users in workflow modelling by proposing extensions of workflows that are under construction. Besides the retrieval of workflows, also their automatic adaptation is recently addressed in research [24, 25].

3 Semantic Workflows

This section provides a focused introduction to semantic workflows. In particular, we describe our graph-based approach for representation as required for the retrieval phase of POCBR.

3.1 Workflows

A workflow is an executable description of a work process that typically involves several persons and/or resources. It consist of a set of *activities* (also called *tasks*) combined with *control-flow structures* like sequences, parallel (AND) or alternative (XOR) branches, as well as repeated execution (LOOPs). Tasks and control-flow structures form the *control-flow*. In addition, tasks exchange certain *data items*, which can also be of physical matter, depending on the workflow domain. Tasks, data items, and relationships between the two of them form the *data-flow*. Today, various workflow languages are used, depending on the kind of workflow. Languages for business workflows (for example BPMN) have a strong focus on the control-flow, while scientific workflow languages have a stronger focus on the data-flow.

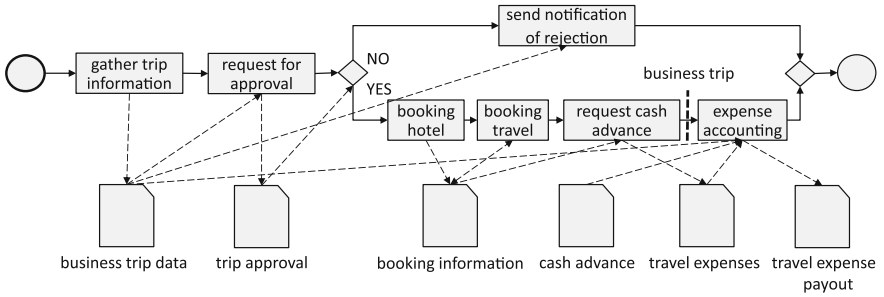

Fig. 1 Examples of a business tip workflow

Figure 1 shows an example of a business workflow. The workflow describes a simple process for a business trip starting with gathering information on the trip and ending with the final expense accounting. It consists of several activities such as booking of the travel or requesting cash advance, which are aligned in sequences (also referred to as control-flow). Furthermore, information is shared between those activities, e.g., business trip data first has to be gathered and then serves as the basics of decision-making whether a trip is approved. Based on the trip approval, the example workflow only executes one specific branch by use of a xor-split/join control-flow pattern.

3.2 Introduction to Semantic Workflows

Traditional workflow languages name task and data items by simple textual labels. This makes it difficult to reason with the knowledge captured in such workflows, as required in POCBR. To address this issue, we introduce *semantic workflows* as an approach to cover relevant aspects of the meaning of workflows, including the meaning of the task and data items that occur. Semantic workflows are based on a specifically designed *domain ontology* that consist of sub-ontologies describing the relevant properties of the task and data items that occur in the domain. A traditional workflow is turned into a semantic workflow by adding metadata annotations from the domain ontology to the individual elements of the workflow. The semantic annotations of workflows can then be used as basis for the similarity assessment and adaptation.

The workflow representation in the CAKE system (see Sect. 6) uses an object-oriented representation for ontologies (classes and relations/properties and inheritance) and metadata annotation (linked instances of classes). Thus, tasks and data items can be organized in a hierarchy of classes, in which each item contains certain properties, which can be inherited from the super class. For example, in Fig. 1, the *gather trip information* task includes a property capturing the employee who is assigned to perform this task.

3.3 Representation of Semantic Workflows

In line with previous work on graph-based workflow representation [26, 27], we represent workflows as semantically labeled directed graphs. A *semantic workflow graph* W is a quadruple $W = (N, E, S, T)$ where N is a set of nodes and $E \subseteq N \times N$ is a set of edges. $T : N \cup E \rightarrow \Omega$ associates to each node and each edge a *type* from Ω (see below). $S : N \cup E \rightarrow \Sigma$ associates to each node and each edge a *semantic description* from the domain ontology Σ. A semantic workflow graph contains the following types of nodes (Ω): Each workflow consists of exactly one *workflow node*. Each task in a workflow is represented by a *task node*. Each data item in a workflow is represented by a *data node*. Each control-flow element in a workflow, such as split/join elements for and/xor blocks, are represented as a *control-flow node*. In addition, a semantic workflow graph contains the following types of edges: *part-of edges* for linking the workflow node to every other node, *data-flow edges* which link data nodes to task nodes or vice versa, and *control-flow edges* connecting two task nodes or a task node with a control-flow node. Figure 2 shows the semantic graph representation of the workflow from Fig. 1. It contains one workflow node, which links all elements of the workflow by part-of edges. A simplified fraction of some semantic descriptions are shown in dashed boxes. The business trip data, for example, could include information about the date, venue or expected expenses of the business trip. Further, gather trip information is annotated by information such as the assigned employee or the task's enactment status.

A fraction of the domain ontology Σ for this example domain is illustrated in Fig. 3. The sub-ontology for tasks is shown, which is a light-weight ontology that

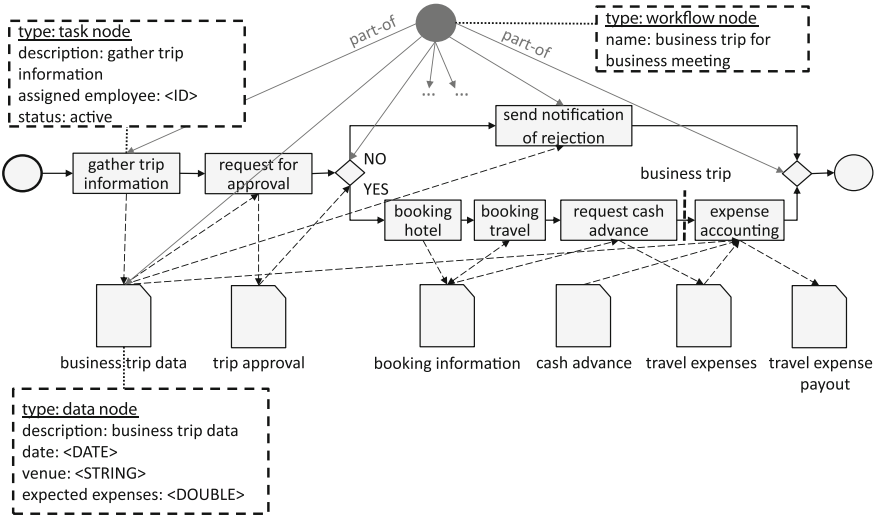

Fig. 2 Semantic workflow of the previously introduced example workflow

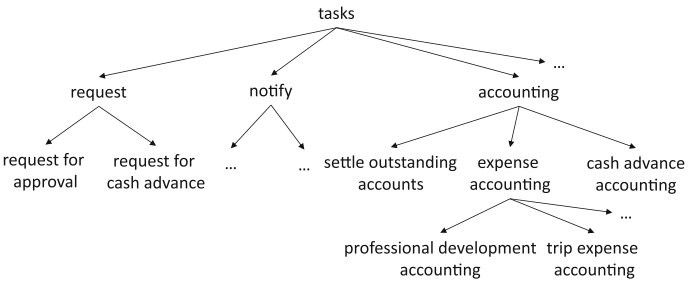

Fig. 3 Example task ontology

arranges relevant tasks in a taxonomical structure. For example, *settle outstanding accounts*, *expense accounting* and *cash advance accounting* are tasks descriptions classified as general *accounting* tasks. Further, *professional development expense accounting* or *trip expense accounting* are special forms of *expense accounting*. In addition, each concept in the ontology may have certain properties (e.g. *assigned employee* in Fig. 2). We employ this ontology for similarity assessment of tasks and data objects (see Sect. 4).

3.4 Repository of Semantic Workflows

A repository of workflows reflecting best-practice processes in a domain is the core of a POCBR approach. In CBR terminology, this repository is the case base that stores the available experience. Given the semantic workflow graph representation, we represent a workflow repository as a set of semantic workflows over the same domain ontology. Hence, a workflow repository is always tied to a particular domain and its semantic metadata representation. Thus, we define a case base $CB = \{CW_1, \ldots, CW_n\}$ as a set of cases CW_i each of which is a semantic workflow graph over Σ.

The acquisition of semantic workflows is an important step in the development of a POCBR system. Usually, workflows are already available in a domain (otherwise the use of POCBR is not advisable), but they have to be captured and formalized as part of a knowledge engineering process. For details, see Sect. 6.3.

4 Similarity-Based Retrieval of Workflows

When using POCBR to support the construction of a new workflow, the user has to specify a query stating the requirements on the workflow s/he is aiming at. The CBR approach then selects the most similar semantic workflow from the repository to that query and adapts it in order increase the similarity to the query.

4.1 User Queries

Users can query a repository of semantic workflows with a specification of important properties of the workflow s/he wants to create in order to retrieve potentially reusable workflows. Previous work on workflow reuse and discovery has already addressed the question how typical queries look like. For scientific workflows, Goderis et al. [28] studied the importance of different criteria for workflow discovery. They identified that the task and data items that occur in the workflow are relevant as well as general characteristics of the workflow related to quality, performance, and reliability. The workflow structure is also very important, particularly the data-flow and control-flow as well as the used control-flow constructs. Also for business workflows, the relevant types of queries have been identified and different query languages have been proposed by Beeri et al. [29] and Awad [30]. Their work clearly shows that it is useful to construct queries in the same way as workflows are constructed. Queries can be patterns built from connected workflow elements, which are then matched against the workflows in the repository.

In the light of these results, we focus on queries that are represented as semantic workflow graphs [31]. Such a query specifies some workflow elements together with the semantic description that are considered as requirements on the workflows to be retrieved. A simple query could even consist solely of a workflow node that contains a semantic description specifying the class of workflow and some general properties, such as quality requirements. More sophisticated queries may in addition contain some unconnected data and/or task nodes that specify that these nodes should occur in the workflow one is looking for. Structural properties related to the data and/or control-flow can be specified by linking the nodes with control-flow and/or data-flow links, thus forming a partial workflow (or even several unconnected partial workflows). The aim of retrieval is then to find workflows from the repository that match the specified partial workflows as good as possible according to a similarity measure.

4.2 Semantic Workflow Similarity

The notion of similarity plays a key role in CBR, since cases are selected based on their similarity to the current query. While early CBR approaches were usually restricted to syntactic similarity measures (such as inverse Euclidean or Hamming distances), the current view is that the similarity measure should consider as much semantics as possible. Consequently, similarity measures must be modeled as part of the knowledge acquisition process during CBR application development.

Similarity is formalized as a function that maps a pair of problem descriptions to a real number, often restricted to the unit interval [0, 1], with the convention that a high value indicates a high similarity. Further, similarity is linked with the notions of preference and utility [32–34]: A higher degree of similarity suggests that a case is more useful for solving the current problem and hence should be preferred.

As a means for practical modeling of similarity functions, the so-called *local-global principle*, first proposed by Richter (for details see [15, 32, 33]) is widely used. Modeling similarity means decomposing the similarity function according to certain properties of the case. A *local similarity function* is defined for each individual property, reflecting the utility of a case with respect to the single property only. The local similarities are then aggregated into the *global similarity* by means of an *aggregation function*. This function appropriately combines all local similarity values (e.g. by a weighted average) into a single value that aims at reflecting the utility of the case as a whole. The graph-based representation of workflows introduces structural aspects into the representation and thereby makes the similarity assessment much more complicated. Several graph algorithms have been proposed for similarity assessment such as sub-graph isomorphism, maximal common sub-graphs, or edit-distance measures [27, 35–38].

In our research, we have developed a new similarity model that is an enhancement of the local-global principle [39]. The local similarity measures assess the similarity between two nodes or two edges of the same type based on their semantic description. The global similarity for workflows is obtained by an aggregation function combining the local similarity values within a graph mapping process.

In more detail, the core of the similarity model is a local similarity measure for semantic descriptions $sim_\Sigma : \Sigma^2 \rightarrow [0, 1]$. In our example domain the taxonomical structure of the data and task ontology is employed to derive a similarity value that reflects the closeness in the ontology. It is combined with additional similarity measures that consider relevant attributes (see [39] for more details and examples).

The similarity $sim_N : N^2 \rightarrow [0, 1]$ of two nodes and two edges $sim_E : E^2 \rightarrow [0, 1]$ is then defined based on sim_Σ applied to their assigned semantic descriptions. The similarity $sim(QW, CW)$ between a query workflow QW and a case workflow CW is defined by means of an admissible mapping $m : N_q \cup E_q \rightarrow N_c \cup E_c$, which is a type-preserving, partial, injective mapping function of the nodes and edges of QW to those of CW. For each query node and edge x mapped by m, the similarity to the respective case node or edge $m(x)$ is computed by $sim_N(x, m(x))$ and $sim_E(x, m(x))$, respectively. The overall workflow similarity with respect to a mapping m, named $sim_m(QW, CW)$ is computed by an aggregation function (e.g. a weighted average) combining the previously computed similarity values. The overall workflow similarity is determined by the best possible mapping m

$$sim(QW, CW) = \max\{sim_m(QW, CW) \mid \texttt{admissible map m}\}.$$

Thus, similarity assessment is defined as an optimization problem that consists of finding the best possible alignment of the query workflow with the case workflow. It determines the best possible way (in terms of similarity) in which the query workflow is covered by the case workflow. In particular, the similarity is 1 if the query workflow is exactly included in the case workflow as a subgraph.

This similarity assessment is then used to retrieve the best matching workflow from the repository. While similarity computation by exhaustive search guarantees to find the optimal match, it is computationally not feasible. This is particularly true

for retrieval of the best matching workflow in large case bases, since the similarity between the query and each case in the case base must be computed. In our research, we developed four different approaches for an efficient retrieval of workflows, which are briefly summarized below.

4.3 Efficient Similarity Computation by Heuristic Search

In a first step, we improved the efficiency of the similarity computation by developing an A* search algorithm, which is based on a specific well-informed admissible heuristic function [39]. The search algorithm aims at finding an admissible map m between the nodes and edges of the workflows to be compared. In the search space the search nodes represent partial maps, which are incrementally extended towards a complete admissible map. As in traditional A*-search [40], in each search step, the search node with the best (in our case the highest) value for the function $f(node) = g(node) + h(node)$ is selected. Here g represents the similarity value already achieved by search node's mapping and h represents an admissible heuristic function providing a good over-estimation of the additional similarity increment that can be achieved by mapping the workflow elements that are not mapped already. With a memory-bound version of A* we achieved a significant speed-up (up-to several orders of magnitude) in similarity computation over exhaustive search while only slightly compromising the precision of the result.

4.4 Parallelized Similarity Computation

An improved version of the presented A* search algorithm results from parallelizing the similarity computations of several (or all) cases of the case base in order to find the k most similar cases. Therefore the search process is parallelized, maintaining one search queue for each case. In every step, the search node from the queue with the highest f-value from all queues is expanded. Search terminates, when at least k searches have terminated and when the similarity of the k-best case is higher than all f-values of the remaining queues. Since the f-values are upper bounds for the final similarity, it is ensured that none of the remaining cases can ever exceed the similarity of the known k-best case. Hence, completeness of k-best retrieval is guaranteed. This approach can also be executed using parallel threads on multi-core CPUs. In our experiments, this approach again leads to a speed-up compared to the A* up-to an order of magnitude for case bases with a few hundred cases and values of $k < 10$.

4.5 Two-Step Retrieval

If the size of the case base is further increased, additional approaches to speed-up retrieval are required. For this purpose a two-level retrieval method has been developed [41] inspired by the MAC/FAC (Many are called, but few are chosen) model originally proposed by Gentner and Forbus [42]. The first retrieval step (MAC phase) performs a rough and efficient pre-selection of a small subset of cases from a large case base. Then, the second step (FAC phase) is executed to perform the computationally expensive graph-based similarity computation on the pre-selected cases only. This method improves the retrieval performance, if the MAC stage can be performed efficiently and if it results in a sufficiently small number of pre-selected cases. However, there is a risk that the MAC phases introduces retrieval errors, as it might disregard highly similar cases due to its limited assessment of the similarity. Hence, the retrieval approach for the MAC phase must be carefully designed such that it is efficient and sufficiently precise in assessing the similarity. We address this problem by proposing an additional feature-based case representation of workflows, which simplifies the original representation while maintaining the most important properties relevant for similarity assessment. This representation is automatically derived from the original graph-based representation. The MAC stage then selects cases by performing a similarity-based retrieval considering the simplified workflow representation.

4.6 Cluster-Based Retrieval

As alternative approach to efficient retrieval we explored the idea to cluster the workflows using a hierarchical clustering algorithm employing the described similarity measure for assessing the distance of two workflows [43]. Therefore, we developed a hierarchical version of the traditional *Partitioning Around Medoids* (PAM) algorithm [44]. It constructs a cluster-tree where each cluster is represented by a particular case (medoid) such that the case base is partitioned into sets of similar cases. This cluster-tree is then used as an index structure during retrieval.

For retrieving the k most similar cases, clusters at predefined levels in the tree are selected that are most similar to the query. Therefore the similarity between the query and a cluster of cases is computed based on the similarity between the query and the medoid representing the cluster. Only the cases within the selected clusters are then considered for similarity-based retrieval. Our investigation revealed that this approach can decrease the retrieval time without a considerable loss of retrieval quality. Furthermore, parameters enable to control the trade-off between retrieval quality and retrieval time. A significant advantage compared to the MAC/FAC approach is that no additional retrieval phase with a separate simplified feature representation must be designed and thus the development and maintenance effort for retrieval is not increased.

5 Workflow Adaptation and Learning Adaptation Knowledge

As introduced in the previous sections, similarity-based retrieval of semantic workflows for a given user query is an important first step to support the reuse of best-practice workflows from a repository. However, finding a similar workflow does not guarantee that it perfectly matches the query. Several requirements stated in the query (see Sect. 4.1) might not be fulfilled by the most similar workflow. Consequently, more or less comprehensive modifications of the workflow are required. To support the user in performing such modifications, an automated workflow adaptation approach is desirable. The overall aim of automated adaptation is to modify the retrieved workflow in such a way that its original similarity to the query is further increased. Ideally, if the adaptation is able to consider all user requirements perfectly, a similarity of 1 is achieved.

In general, adaptation methods in CBR can be roughly classified into transformational, compositional, and generative adaptation [45]. While transformational adaptation (e.g., [46]) relies on adaptation knowledge represented as rules or operators, generative adaptation demands general domain knowledge appropriate for an automated from scratch problem solver. In compositional adaptation several components from various cases are reused during adaptation, incorporating transformational or generative adaptation methods. Also generalization of cases can be used for the purpose of adaptation, since a single generalized case (e.g., [47, 48]) comprises adaptation knowledge, which provides solutions for a range of problems. Thus, all adaptation approaches require some kind of adaptation knowledge in the particular application domain, for example, in the form of rules describing the replacement of domain-specific tasks or data items. However, the acquisition of such adaptation knowledge is a complex and laborious task. This results in a knowledge-acquisition bottleneck of adaptation knowledge [49], impeding successful workflow adaptation. Thus, various approaches have been proposed to learn adaptation knowledge automatically [16, 50, 51].

In our work, we developed various adaptation approaches for POCBR in order to support individual workflow reuse. As we aim at avoiding the acquisition bottleneck for adaptation knowledge, for each adaptation method described in the following, a specific approach for learning the required adaptation knowledge is presented. This adaptation knowledge is determined prior to retrieval and adaptation from the workflows stored in repository.

5.1 Adaptation by Generalization and Specialization

Generalization and Specialization is an adaptation approach in CBR, in which the adaptation knowledge is learned by a generalization of the cases. The generalized cases are then stored in the case base. Thus, for a particular problem scenario a gener-

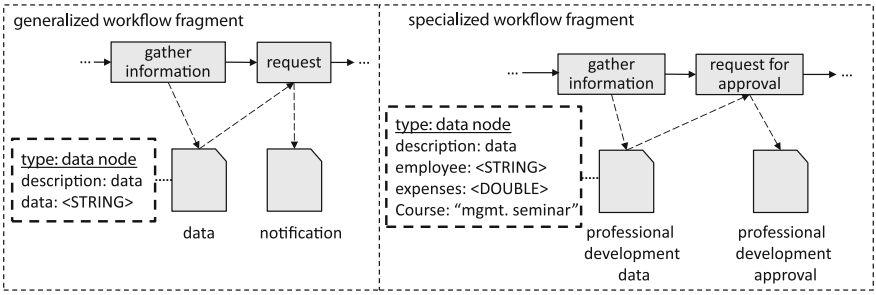

Fig. 4 Example of a generalized and a corresponding specialized workflow

alized case can be retrieved from the case base and refined according to the problem at hand. This approach can be easily applied to workflow cases. A generalized workflow [52] is a workflow whose elements (task and data items) are described by generalized terms from the domain ontology (see Sect. 3). Each generalized term represents multiple specific task or data objects. Thus, a generalized workflow represents several specialized workflows.

An example is sketched in Fig. 4, illustrating a generalized workflow fragment and a corresponding specialized workflow fragment. The generalized fragment can be derived from the example workflow given in Fig. 2. It describes that a request requires the gathering of some data prior to a notification, thereby making no assumption on the concrete request or data present. Thus, it is a generalized process fragment that can be used for many scenarios, for example, the approval for professional development as illustrated in the corresponding specialized workflow fragment. This process fragment involves particular task and data items suited to the particular professional development scenario. A specialized workflow consequently represents the entire process for a concrete scenario. Please note that specialization usually involves information at different levels of abstraction, i.e., professional development data involves information on the particular course, expenses, or related employee, while general data makes no concrete assumption on the information given. Thus, specialization of tasks or data items may also result in a replacement of the entire semantic description (see attached boxes).

A generalized workflow can be learned by comparing similar workflows from the repository (see [52] for technical details). This approach is based on the assumption that if similar workflows contain similar terms, these terms can be replaced by a more generalized term from the ontology. This aims at learning only reasonable generalizations in order to ensure adaptation quality. For example, the generalized workflow data item *data* illustrated in Fig. 4, could result from the fact that similar workflows contain the terms *professional development data, business trip data*, and *business meeting data*. Then the illustrated workflow can be generalized to contain any kind of *data*. Accordingly, the generalized task *request* represents all possible kinds of requests.

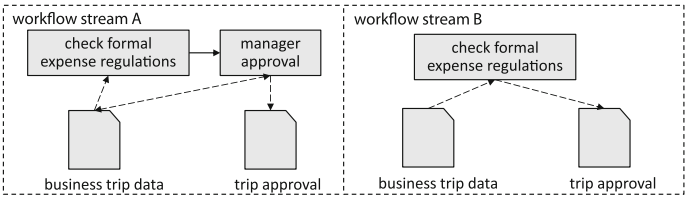

Fig. 5 Example of two workflow streams A and B

Adaptation is supported by specializing a workflow according to the query. This means that each generalized task or data item is replaced by a specialized node such that the similarity between the query and the adapted workflow is maximized. For example, if the generalized workflow contains the term *data* and the query defines that *business meeting data* is desired, then the generalized element description is specialized to *business meeting data*. If a generalized workflow covers several specialized workflows, the workflow repository size can be reduced. This simplifies the repository management and increases the workflow retrieval performance.

5.2 Compositional Adaptation

The idea of compositional adaptation [53] is that each workflow can be decomposed into meaningful sub-workflows. This decomposition is based on the fact that the final workflow output is quite often achieved by producing partial outputs that are somehow combined to create the final workflow output. Partial outputs are generated by particular parts of the workflow (sub-graphs), which we refer to as *workflow streams*. For example, the task *request for approval* illustrated in Fig. 2 could be alternatively performed by the workflow streams illustrated in Fig. 5. These workflow streams describe two alternative decision processes for tip approval. Stream A may refer to a business domain in which a manager approval is required, while stream B could represent a particular process in a government domain as it merely focuses on the formal expense regulations. In general, there can be many workflow streams that could potentially be exchanged with one another to achieve the same partial output.

The basic idea for compositional adaptation is to adapt a workflow by replacing workflow streams in the retrieved workflow with more suitable workflow streams from other workflows. This means that the workflow streams represent the required adaptation knowledge for compositional adaptation. Thus, prior to adaptation, useful workflow streams are extracted from the workflows stored in the repository. Workflow streams can be identified by collecting all data-flow connected tasks[1] until a new data item is created, denoting the corresponding partial workflow output (see for [53] details).

[1] if a tasks consumes a data item produced by another one, they are data-flow connected.

In order to adapt a workflow, a workflow stream can be replaced by a stream learned from another workflow that produces the same partial output but in a different manner, i.e., with other task or data items. Workflow streams can only be replaced, if their data nodes indicate that they represent the same kind of sub-process. This ensures that replacing an arbitrary stream does not violate the correctness of the workflow. In the given example, workflow streams can only be replaced if they consume a data item *business trip data* and produce a *trip approval* data item.

The overall compositional adaptation process aims at increasing the similarity between the query and the adapted workflow by successively replacing workflow streams. Each workflow stream is replaced by the respective stream, which maximizes the overall workflow similarity. The resulting adapted workflow is thus a local optimum achieved by adapting the retrieved most similar workflow using the available workflow streams.

5.3 Transformational Adaptation

Transformational adaptation is based on adaptation knowledge in the form of adaptation rules or adaptation operators [45] that specify a particular modification of the case. Our transformational approach to workflow adaptation focuses on workflow adaptation operators [54] which are specified in a STRIPS-like manner. An operator consists of two workflow sub-graphs we call *streamlets*: a DELETE-streamlet specifies a workflow fraction to be deleted from the workflow and an ADD-streamlet represents a workflow fraction to be added to the workflow. Thereby operators can define the insertion, the removal, or the replacement of a particular workflow fraction. In contrast to compositional adaptation, not only workflow streams can be replaced, but basically any fraction of a workflow, such as a single task or a single data item.

The example adaptation operator shown in Fig. 6 describes the transformation of a planned business trip towards a spontaneous short time customer meeting near-by. Thus, the booking of hotel and travel as well as the request for cash advance becomes superfluous. Instead, just a taxi would have to be ordered. This change does not only affect the activities but also the data items of the workflow, since here the data item *booking information* is no longer required.

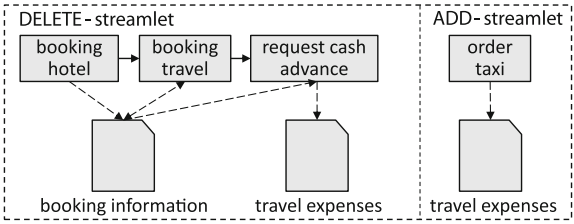

Fig. 6 Example of a workflow adaptation operator

Workflow adaptation operators can be learned from the workflow repository by analyzing pairs of highly similar workflows (selected by using a similarity threshold). For each pair, the difference is determined and workflow operators are generated, whose ADD and DELETE-streamlets basically cover those differences. Roughly speaking, the generated operators thus transform one workflow of the pair into the other one (see [54] for a detailed description of the algorithm).

For adaptation, the learned adaptation operators are applied using a local search algorithm, in a similar manner as in compositional adaptation. The resulting adapted workflow is thus a local optimum achieved by adapting the retrieved most similar workflow using the available adaptation operators.

The three described adaptation methods can also be combined to a single adaptation process (see [55] for details). This comprehensive adaptation, integrates and combines all adaptation methods, thereby maximizing the opportunity to generate a suitable workflow for the given query.

6 CAKE - An Integrated System for Process-Oriented Case-Based Reasoning

In the following, we briefly sketch the CAKE framework, which includes the previously introduced methods and we highlight several application examples.

6.1 Achitecture

The CAKE[2] architecture [13] (see Fig. 7) basically consist of data bases as well as a client and a server component. The latter includes a storage layer which handles persistence of all data, an interface layer for client communication and two central engines, i.e., the agile workflow engine and the knowledge engine working together on the same data items accessed via the storage layer. CAKE is implemented in JAVA as Web-based system running as a Software as a Service.

The *agile workflow engine* is used for the enactment of agile workflows and supports their collaborative modeling. Furthermore, changes on demand can be collaboratively performed on workflow instances as well as workflow models at any time. Running workflow instances delegate tasks to humans via the worklist manager and applications may be invoked via the service connector. In this chapter, however, we mainly focused on the *knowledge engine*, which supports users in finding, defining, and adapting workflows according to their current needs. Therefore, the knowledge engine implements the retrieval and adaptation of semantic workflows as previously introduced. CAKE ensures that any stored resource (a workflow, a task, a document, and any further workflow related resources) is accessible and possesses a clear

[2] See http://cake.wi2.uni-trier.de.

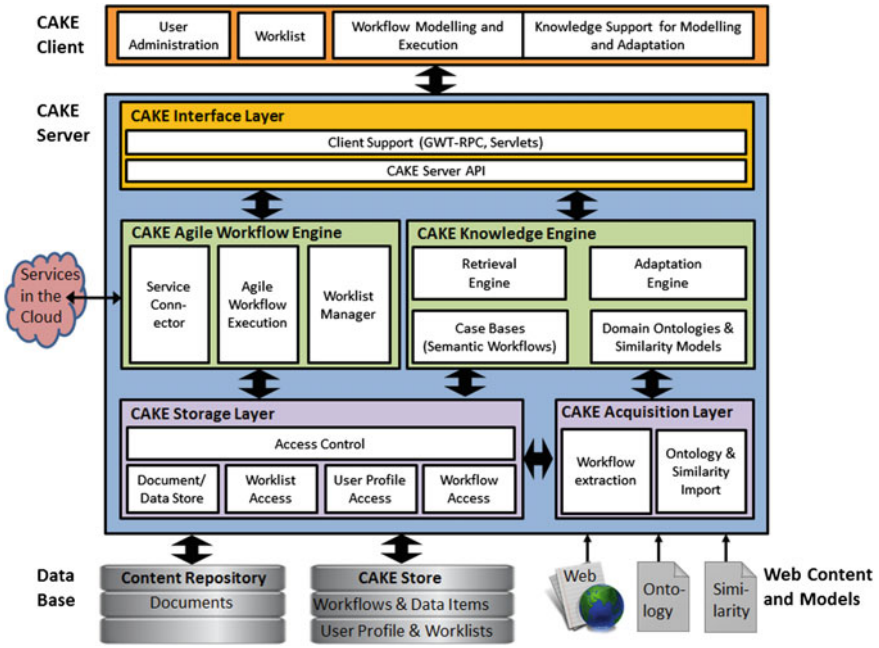

Fig. 7 CAKE system architecture

ownership by means of the resource model implemented in the *storage layer*. This way, workflows can be shared and reused considering particular access rights [56]. The *acquisition layer* handles the import of ontologies and similarity measures and further supports the automatic extraction of workflows from text [57, 58]. CAKE provides two client interfaces, i.e., a browser based access as well as a mobile android application, which are connected to the server component via the *interface layer*.

6.2 Selected Application Examples

The methods for supporting workflow reuse presented in this chapter have potential to be useful in many application areas, some of which will be now briefly sketched.

From the traditional business perspective, the presented methods can support the creation of *business processes* addressing the individual needs of customers or of a particular business scenario. Furthermore, adaptation enables the automatic modification of workflow models, for example, to suit changed business environments. Moreover, POCBR can be employed as a knowledge management method for capturing, storing, and sharing procedural knowledge among different employees or departments.

Social Workflows are a new research area [59, 60] which addresses the support of processes enacted during daily life, such as, do-it-yourself car repair, moving to another city, or organizing a trip with a group of friends. The steps in a social workflow involve access to social networks, the activation of online services, as well as activities performed by several people (e.g., friends or professionals). In a social workflow management system as introduced by Görg [59, 60], the reuse capabilities illustrated in this chapter are highly relevant since the users of social workflows are not experienced in workflow modeling.

In the *cooking domain* a recipe can also be represented as a workflow describing the instructions for cooking a particular dish [55]. While traditional recipe websites solely regard ingredients, categories or recipe names during recipe search, CAKE is able to consider additional knowledge such as required cooking steps, difficulty level, costs, resource consumption, available tools, and diaries. Cooking workflows can be selected and adapted considering particular user preferences by employing the previously introduced POCBR methods within the knowledge engine. Subsequently, CAKE provides a step-by-step guidance for the preparation of the particular dish.

6.3 Required Knowledge Engineering

In order to apply POCBR (including the methods described in this chapter) in a certain domain requires a knowledge engineering process. This knowledge engineering involves the development of the ontology (including task and data sub-ontologies), the similarity measures, as well as the workflows for the case base. The manual acquisition of adaptation knowledge is not required as this knowledge is obtained by the described machine learning approaches particularly targeting the knowledge required for the various adaptation methods.

Ontology development can be performed according to standard methodologies (see, for example [61], Chap. 4 for an introduction). In particular, the reuse of existing ontologies is highly recommended. In the cooking domain, for example, we make use of the cooking ontology developed within the Taaable project[3] [62], which already includes a huge set of ingredients and cooking steps. The sub-ontologies we currently use consist of 156 ingredients and 55 cooking steps.

Based on the ontology, local similarity measures for comparing task and data items must be developed. The knowledge engineering process for similarity measures is well established in CBR [63] and ends-up in selecting appropriate local measures from a similarity-measure library and selecting their parameters according to the domain of the attributes (see, for example [33], Chap. 4).

The final knowledge-engineering steps aims at the acquisition of semantic workflows to populate the case base. For this purpose, different approaches are possible.

[3] http://wikitaaable.loria.fr.

- Workflows can be acquired in a manual process by using workflow modeling tools. Then, appropriate semantic annotations must be added to each task and data item. This manual process is obviously a quite laborious activity, but it ensures a high quality of the resulting knowledge.
- Alternatively, already existing workflows represented in some standard format, such as BPMN, can be reused. For example, existing workflow repositories collected and published for research purposes can be a good starting point. Current collections include the BPM Academic Initiative Model Collection,[4] selected reference models from the IBM Academic Initiative program,[5] the SAP reference models [64], the collections used in the process model matching contests[6] in 2013 and 2015 as well as the collection of the Institute for Information Systems at the DFKI in Saarbrücken [65]. However, the models in these collections lack the required semantic descriptions and thus also requires an additional annotation process before being usable as cases.
- The third approach for acquiring workflows is to apply information extraction methods on available textual descriptions of processes [57, 58, 66]. Cooking recipes, for example, are appropriate textual descriptions. The preparation instructions for the dish included in a recipe can be analyzed and turned into a formal workflow representation. Also in other domains, workflows are described in a textual fashion, for example procedures for technical diagnosis. However, the resulting workflows still need manual quality control and improvement as automatic methods are yet unable to produce results with sufficient quality.

From our experience, the described knowledge-engineering approach is appropriate and the effort is acceptable in many domains. As benefit from these development efforts, POCBR comprehensively supports the creation of new workflows by reuse. Of course, there is no guarantee that workflows that are produced as a result of an adaptation process are always semantically correct (the syntactic correctness, however, is guaranteed). Their correctness depends on the correctness of the learned adaptation knowledge. As this learning process is an inductive process, its correctness cannot be ensured. Thus, there is always a need for a human user to access and validate the resulting adapted workflows.

7 Conclusions

In this chapter, we introduced process-oriented case-based reasoning as a method to support flexible and more individual workflows. We presented an overview of different methods from knowledge representation, knowledge engineering and machine learning to support the representation of semantic workflows as well as their simi-

[4] http://bpmai.org/download/index.html.

[5] https://developer.ibm.com/academic.

[6] https://ai.wu.ac.at/emisa2015/contest.php.

larity based retrieval and adaptation. These methods have been demonstrated using an example from a traditional business process, involving several manual activities typically enacted with support by some specific business application software. Thus, those workflows are a means to coordinate the work among the involved employees but they are also a means for application integration. As pointed out in the introduction, workflows can also used in several other application contexts. Here, the spectrum is quite large, leading applications involving a flow of activities performed completely manual (such as in the cooking domain) up to applications in which the workflows are executed fully automatically (e.g. scientific workflows or workflows for information integration).

Within the scope of our work, we have extensively evaluated the proposed methods, including the quality of the adapted workflows in the cooking domain [39, 52–54]. Initial experimental evaluations in the domain of business processes [67], and social workflows [59, 60] have been performed as well. Future work will focus on investigating these and other new application domains in more depth. Moreover, we will investigate interactive methods, which involve the user during search and adaptation of workflows to further enhance the usability of the presented methods.

Acknowledgements This research was funded in part under grant BE1373/3-1 from the German Research Foundation (DFG) and enacted in cooperation with Mirjam Minor and her group from Goethe University Frankfurt.

References

1. van der Aalst, W.M.: Business process management: a comprehensive survey. ISRN Softw. Eng. **2013**, 1–37 (2013)
2. Workflow Management Coalition: Workflow management coalition glossary & terminology (1999)
3. Freßmann, A., Sauer, T., Bergmann, R.: Collaboration patterns for adaptive software engineering processes. In: Czap, H., Unland, R., Branki, C., Tianfield, H. (eds.) Self-Organization and Autonomic Informatics (I), vol. 135, pp. 304–312. IOS Press, Amsterdam (2005). ISBN 1-58603-577-0
4. Minor, M., Tartakovski, A., Schmalen, D., Bergmann, R.: Agile workflow technology and case-based change reuse for long-term processes. International Journal of Intelligent Information Technologies **4**(1), 80–98 (2008)
5. Taylor, I.J., Deelman, E., Gannon, D.B.: Workflows for e-Science. Springer, Berlin (2007)
6. Freßmann, A.: Adaptive workflow support for search processes within fire service organisations. In: Reddy, S.M. (ed.) Proceedings of the Fifteenth IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 291–296. IEEE Computer Society (2006)
7. Hung, P., Chiu, D.: Developing workflow-based information integration (WII) with exception support in a web services environment. In: Proceedings of the 37th Annual Hawaii International Conference on System Sciences, 2004, p. 10 (2004)
8. Minor, M., Bergmann, R., Görg, S., Walter, K.: Adaptation of cooking instructions following the workflow paradigm. In: Marling, C. (ed.) ICCBR 2010 Workshop Proceedings (2010)
9. Fleischmann, A., Schmidt, W., Stary, C., Augl, M.: Agiles prozessmanagement mittels subjektorientierung. HMD Praxis der Wirtschaftsinformatik **50**(2), 64–76 (2013)

10. Reichert, M., Weber, B.: Enabling Flexibility in Process-aware Information Systems: Challenges, Methods, Technologies. Springer Science & Business Media, Berlin (2012)
11. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Commun. **7**(1), 39–59 (1994)
12. Minor, M., Montani, S., Recio-García, J.A.: Process-oriented case-based reasoning. Inf. Syst. **40**, 103–105 (2014)
13. Bergmann, R., Gessinger, S., Görg, S., Müller, G.: The collaborative agile knowledge engine CAKE. In: Goggins, S.P., Jahnke, I., McDonald, D.W., Bjørn, P. (eds.) Proceedings of the 18th International Conference on Supporting Group Work, Sanibel Island, FL, USA, November 09–12, 2014, pp. 281–284. ACM (2014)
14. Richter, M.M., Weber, R.O.: Case-Based Reasoning - A Textbook. Springer, Berlin (2013)
15. Lopez De 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. Knowl. Eng. Rev. **20**(3), 215–240 (2005)
16. Craw, S., Wiratunga, N., Rowe, R.C.: Learning adaptation knowledge to improve case-based reasoning. Artif. Intell. **170**(16), 1175–1192 (2006)
17. Badra, F., Cordier, A., Lieber, J.: Opportunistic adaptation knowledge discovery. In: McGinty, L., Wilson, D.C. (eds.) Case-Based Reasoning Research and Development, 8th International Conference on Case-Based Reasoning, ICCBR 2009. Lecture Notes in Computer Science, vol. 5650, pp. 60–74. Springer, Berlin (2009)
18. Dufour-Lussier, V., Ber, F.L., Lieber, J., Nauer, E.: Automatic case acquisition from texts for process-oriented case-based reasoning. Inf. Syst. **40**, 153–167 (2014)
19. Leake, D.B., Wilson, D.C.: Combining CBR with interactive knowledge acquisition, manipulation and reuse. In: Proceedings of the Third International Conference on Case-Based Reasoning and Development. ICCBR '99, pp. 203–217. Springer, London (1999)
20. Bergmann, R., Freßmann, A., Maximini, K., Maximini, R., Sauer, T.: Case-based support for collaborative business. In: Proceedings of the 8th European Conference on Advances in Case-Based Reasoning. ECCBR'06, pp. 519–533. Springer, Berlin (2006)
21. Montani, S., Leonardi, G.: Retrieval and clustering for supporting business process adjustment and analysis. Inf. Syst. **40**, 128–141 (2014)
22. Sauer, T., Maximini, K.: Using workflow context for automated enactment state tracking. In: Minor, M. (ed.) Workshop Proceedings: 8th European Conference on Case-Based Reasoning, Workshop: Case-based Reasoning and Context Awareness (CACOA 2006), pp. 300–314. Universität Trier (2006)
23. Madhusudan, T., Zhao, J.L., Marshall, B.: A case-based reasoning framework for workflow model management. Data Knowl. Eng. **50**(1), 87–115 (2004)
24. Müller, R., Greiner, U., Rahm, E.: $A_{gent}W_{ork}$: a workflow system supporting rule-based workflow adaptation. Data Knowl. Eng. **51**(2), 223–256 (2004). doi:10.1016/j.datak.2004.03.010
25. Weber, B., Wild, W., Breu, R.: CBRFlow: enabling adaptive workflow management through conversational case-based reasoning. In: Funk, P., González-Calero, P.A. (eds.) Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004, Madrid, Spain, August 30 - September 2, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3155, pp. 434–448. Springer, Berlin (2004). doi:10.1007/978-3-540-28631-8_32
26. Champin, P.A., Solnon, C.: Measuring the similarity of labeled graphs. In: Case-Based Reasoning Research and Development, pp. 1066–1067. Springer, Berlin (2003)
27. Dijkman, R., Dumas, M., Garcia-Banuelos, L.: Graph matching algorithms for business process model similarity search. In: Business Process Management, pp. 48–63 (2009)
28. Goderis, A., Li, P., Goble, C.: Workflow discovery: the problem, a case study from e-science and a graph-based solution. Int. J. Web Serv. Res. **5**(4), 2 (2008)
29. Beeri, C., Eyal, A., Kamenkovich, S., Milo, T.: Querying business processes. In: Proceedings of the 32nd International Conference on Very Large Data Bases, VLDB '06, pp. 343–354. VLDB Endowment (2006)
30. Awad, A.: BPMN-Q: a language to query business processes. In: Reichert, M., Strecker, S., Turowski, K. (eds.) Enterprise Modelling and Information Systems Architectures - Concepts

and Applications. Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA'07), St. Goar, Germany, October 8–9, 2007. LNI, vol. 119, pp. 115–128. GI (2007)
31. Müller, G., Bergmann, R.: POQL: a new query language for process-oriented case-based reasoning. In: Proceedings of the LWA 2015 Workshops: KDML, FGWM, IR, and FGDB. CEUR Workshop Proceedings, vol. 1458, pp. 247–255. Trier (2015). http://www.wi2.uni-trier.de/publications/2015_MuellerBergmannLWA.pdf
32. Richter, M.M.: Foundations of similarity and utility. In: Proceedings of the 20th International Florida Artificial Intelligence Research Society Conference (FLAIRS 2007). AAAI Press (2007)
33. Bergmann, R.: Experience Management - Foundations, Development Methodology, and Internet-Based Applications. LNAI, vol. 2432. Springer, Berlin (2002)
34. Bergmann, R., Richter, M.M., Schmitt, S., Stahl, A., Vollrath, I.: Utility-oriented matching: a new research direction for case-based reasoning. In: Schmitt, S., Vollrath, I., Reimer, U. (eds.) 9th German Workshop on Case-Based Reasoning, pp. 264–274 (2001)
35. Kapetanakis, S., Petridis, M., Knight, B., Ma, J., Bacon, L.: A case based reasoning approach for the monitoring of business workflows. In: Bichindaritz, I., Montani, S. (eds.) Case-Based Reasoning. Research and Development, ICCBR 2010, pp. 390–405. Springer (2010)
36. Montani, S., Leonardi, G., Lo Vetere, M.: Case retrieval and clustering for business process monitoring. In: Proceedings of the ICCBR 2011 Workshops, pp. 77–86 (2011)
37. Goderis, A.: Workflow re-use and discovery in bioinformatics. Ph.D. thesis, University of Manchester (2008)
38. Leake, D.B., Kendall-Morwick, J.: Towards case-based support for e-science workflow generation by mining provenance. In: Althoff, K.D., Bergmann, R., Minor, M., Hanft, A. (eds.) Advances in CBR, pp. 269–283 (2008)
39. Bergmann, R., Gil, Y.: Similarity assessment and efficient retrieval of semantic workfows. Inf. Syst. **40**, 115–127 (2014)
40. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, Englewood Cliffs (2010)
41. Bergmann, R., Stromer, A.: MAC/FAC retrieval of semantic workflows. In: Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2013, St. Pete Beach, Florida. May 22–24, 2013. (2013)
42. Gentner, D., Forbus, K.D.: MAC/FAC: a model of similarity-based retrieval. In: Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society. Cognitive Science Society (1991)
43. Müller, G., Bergmann, R.: A cluster-based approach to improve similarity-based retrieval for Process-Oriented Case-Based Reasoning. In: 20th European Conference on Artificial Intelligence (ECAI 2014), pp. 639–644. IOS Press (2014)
44. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data - An Introduction to Cluster Analysis. Wiley, New York (1990)
45. Wilke, W., Bergmann, R.: Techniques and knowledge used for adaptation during case-based problem solving. In: Pobil, A.P.D., Mira, J., Ali, M. (eds.) 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE-98. Lecture Notes in Computer Science, vol. 1416, pp. 497–506. Springer, Berlin (1998)
46. Minor, M., Bergmann, R., Görg, S., Walter, K.: Towards case-based adaptation of workflows. In: Case-Based Reasoning. Research and Development, pp. 421–435. Springer, Berlin (2010)
47. Maximini, K., Maximini, R., Bergmann, R.: An investigation of generalized cases. In: Ashley, K.D., Bridge, D.G. (eds.) Case-Based Reasoning Research and Development, 5th International Conference on Case-Based Reasoning, ICCBR 2003, Trondheim, Norway, June 23-26, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2689, pp. 261–275. Springer, Berlin (2003). doi:10.1007/3-540-45006-8_22
48. Bergmann, R., Vollrath, I.: Generalized cases: representation and steps towards efficient similarity assessment. In: Burgard, W., Christaller, T., Cremers, A.B. (eds.) KI-99: Advances in

Artificial Intelligence, 23rd Annual German Conference on Artificial Intelligence, Bonn, Germany, September 13–15, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1701, pp. 195–206. Springer, Berlin (1999). doi:10.1007/3-540-48238-5_16
49. Hanney, K., Keane, M.T.: The adaption knowledge bottleneck: how to ease it by learning from cases. In: Leake, D.B., Plaza, E. (eds.) Case-Based Reasoning Research and Development, Second International Conference, ICCBR-97, Providence, Rhode Island, USA, July 25–27, 1997, Proceedings. Lecture Notes in Computer Science, vol. 1266, pp. 359–370. Springer, Berlin (1997)
50. Minor, M., Görg, S.: Acquiring adaptation cases for scientific workflows. In: Case-Based Reasoning. Research and Development, 19th International Conference on Case-Based Reasoning, ICCBR 2011. Lecture Notes in Computer Science, vol. 6880, pp. 166–180. Springer, Berlin (2011)
51. Hanney, K., Keane, M.T.: Learning adaptation rules from a case-base. In: Smith, I.F.C., Faltings, B. (eds.) Advances in Case-Based Reasoning, Third European Workshop, EWCBR-96, Lausanne, Switzerland, November 14–16, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1168, pp. 179–192. Springer, Berlin (1996)
52. Müller, G., Bergmann, R.: Generalization of workflows in process-oriented case-based reasoning. In: Russell, I., Eberle, W. (eds.) Proceedings of the Twenty-Eighth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2015, Hollywood, Florida, May 18–20, 2015, pp. 391–396. AAAI Press (2015)
53. Müller, G., Bergmann, R.: Workflow streams: a means for compositional adaptation in process-oriented CBR. In: Lamontagne, L., Plaza, E. (eds.) Case-Based Reasoning Research and Development - 22nd International Conference, ICCBR 2014, Cork, Ireland, September 29, 2014–October 1, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8765, pp. 315–329. Springer, Berlin (2014)
54. Müller, G., Bergmann, R.: Learning and applying adaptation operators in process-oriented case-based reasoning. In: Hüllermeier, E., Minor, M. (eds.) Case-Based Reasoning Research and Development, ICCBR 2015, Frankfurt am Main, Germany, September 28–30, 2015, Proceedings. LNCS, vol. 9343, pp. 259–274. Springer, Berlin (2015)
55. Müller, G., Bergmann, R.: CookingCAKE: a framework for the adaptation of cooking recipes represented as workflows. In: Kendall-Morwick, J. (ed.) Workshop Proceedings from The Twenty-Third International Conference on Case-Based Reasoning (ICCBR 2015), Frankfurt, Germany, September 28–30, 2015. *CEUR Workshop Proceedings*, vol. 1520, pp. 221–232. CEUR-WS.org (2015). http://ceur-ws.org/Vol-1520/paper23.pdf
56. Görg, S., Bergmann, R., Gessinger, S., Minor, M.: A resource model for cloud-based workflow management systems - enabling access control, collaboration and reuse. In: Desprez, F., Ferguson, D., Hadar, E., Leymann, F., Jarke, M., Helfert, M. (eds.) CLOSER 2013 - Proceedings of the 3rd International Conference on Cloud Computing and Services Science, Aachen, Germany, 8–10 May, 2013, pp. 263–272. SciTePress (2013)
57. Schumacher, P.: Workflow Extraction from Textual Process Descriptions. Verlag Dr. Hut, München (2016)
58. Schumacher, P., Minor, M., Walter, K., Bergmann, R.: Extraction of procedural knowledge from the web. In: Workshop Proceedings: WWW'12. Lyon, France (2012)
59. Görg, M.S.: Social Workflows, pp. 77–110. Springer Fachmedien Wiesbaden, Wiesbaden (2016)
60. Görg, S., Bergmann, R.: Social workflows vision and potential study. Inf. Syst. **50**, 1–19 (2015)
61. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman and Hall/CRC Press (2010). http://www.semantic-web-book.org/
62. Cordier, A., Lieber, J., Molli, P., Nauer, E., Skaf-Molli, H., Toussaint, Y.: WIKITAAABLE: a semantic wiki as a blackboard for a textual case-base reasoning system. In: 4th Semantic Wiki Workshop (SemWiki 2009) at the 6th European Semantic Web Conference (ESWC 2009), Hersonissos, Greece, June 1st, 2009. Proceedings. CEUR Workshop Proceedings, vol. 464. CEUR-WS.org (2009). http://ceur-ws.org/Vol-464

63. Stahl, A.: Defining similarity measures: Top-down vs. bottom-up. In: Craw, S., Preece, A.D. (eds.) Advances in Case-Based Reasoning, 6th European Conference, ECCBR 2002 Aberdeen, Scotland, UK, September 4–7, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2416, pp. 406–420. Springer, Berlin (2002)
64. Curran, T.A., Ladd, A.: SAP R/3 Business Blueprint: Understanding Enterprise Supply Chain Management. Prentice Hall International, Englewood Cliffs (1999)
65. Thaler, T., Dadashnia, S., Sonntag, A., Fettke, P., Loos, P.: The IWi process model corpus. Technical report, Saarländische Universitäts- und Landesbibliothek, Postfach 151141, 66041 Saarbrücken (2015). http://scidok.sulb.uni-saarland.de/volltexte/2015/6267
66. Schumacher, P., Minor, M.: Extracting control-flow from text. In: Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration, pp. 203–210. IEEE, San Francisco, California, USA (2014)
67. Pfister, M., Fuchs, F., Bergmann, R.: Ähnlichkeitsbasiertes Retrieval von BPMN-2.0-Modellen. In: Lernen, Wissen, Daten, Analysen (LWDA 2016) (2016). http://www.wi2.uni-trier.de/publications/2016_PfisterFuchsBergmann_LWDA.pdf