Published at Computer Cooking Contest 2015, Workshop Proceedings
from The 23rd International Conference on Case-Based Reasoning (ICCBR 2015).
The original publication is available at http://ceur-ws.org/Vol-1520/

CookingCAKE: A Framework for the adaptation of cooking recipes represented as workflows

Gilbert Müller and Ralph Bergmann

Business Information Systems II
University of Trier
54286 Trier, Germany
[muellerg][bergmann]@uni-trier.de,
www.wi2.uni-trier.de

Abstract. This paper presents CookingCAKE, a framework for the adaptation of cooking recipes represented as workflows. CookingCAKE integrates and combines several workflow adaptation approaches applied in process-oriented case based reasoning (POCBR) in a single adaptation framework, thus providing a capable tool for the adaptation of cooking recipes. The available case base of cooking workflows is analyzed to generate adaptation knowledge which is used to adapt a recipe regarding restrictions and resources, which the user may define for the preparation of a dish.

Keywords: recipe adaptation, workflow adaptation, workflows, process-oriented case based reasoning

1 Introduction

Even after more than 30 years of research in CBR, adaptation is still a major challenge. This also applies to the adaptation of cooking recipes. Direct processing of textual recipes is however almost not feasible. Thus, they are usually transformed to structured cases, e.g., workflows [18]. In Process-Oriented Case-Based Reasoning (POCBR) [11], workflow adaptation is also an important research topic.

Existing methods for adaptation in CBR can be roughly classified into transformational, compositional, and generative adaptation [21,9]. While transformational adaptation relies on adaptations executed in a kind of a rule-based manner, generative adaptation demands general domain knowledge appropriate for an automated from scratch problem solver. An approach for transformational adaptation of workflows was presented by Minor et al. [10]. Compositional adaptation usually means that several cases are used during adaptation, incorporating transformational or generative adaptation methods involving adaptation knowledge. Dufour-Lussier et al. [4], for example, presented such a compositional adaptation approach. However, the different adaptation approaches come along with respective advantages and disadvantages. Thus, we expect that the integration and combination of several adaptation approaches can significantly improve the overall adaptation capability of a CBR system by overcoming some of the disadvantages of each individual approach.

In this paper, we present the next evolutionary step of your CookingCAKE system, which is the integration of three adaptation approaches developed within our previous research. In particular, we present a novel integration of adaptation by generalization and specialization, compositional adaptation, and transformational adaptation in POCBR. To achieve this, CookingCAKE analyzes the case base of cooking workflows generating an extensive adaptation knowledge base using several adaptation approaches. This knowledge is then used to adapt workflows according to the requirements and resources given in a current adaptation scenario. While this paper presents novel ideas and positions on adaptation, the *open challenge* is addressed. In addition, we present our examples as well as a comprehensive use case from the *sandwich challenge*, thus this challenge is addressed as well.

The next section introduces cooking workflows followed by a summary section sketching the used adaptation approaches from our previous research (see Sect. 3). Section 4 describes the novel integration of the approaches, including the generation of adaptation knowledge as well as the integrated adaptation itself. Next, Sect. 5 provides details on how the Computer Cooking Contest 2015 sandwich challenge is addressed. Finally, the paper wraps up by discussing potential future work.

2 Cooking Workflows

In our approach a cooking recipe is represented as a workflow describing the process to prepare a particular dish [18] (see Fig. 1). Cooking workflows consist of a set of *preparation steps* (also called *tasks*) and a set of *ingredients* (also called *data items*) shared between its tasks. Further, control-flow blocks may be used that represent either sequences, parallel (AND), alternative (XOR), or repeated execution (LOOPs) of preparation steps. These control-flow blocks may be nested but not interleaved, thus we consider block-oriented workflows only. This ensures the syntactic correctness of the workflow following the correctness-by-construction principle [17,3], e.g., that the workflow has one start node and one end node. Such workflows are referred to as consistent workflows. Tasks and control-flow blocks are linked by *control-flow edges* defining

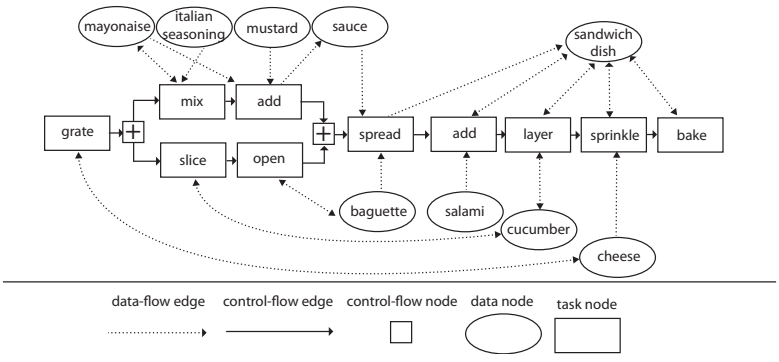

Fig. 1. Example of a block-oriented cooking workflow

the execution order. This forms the *control-flow*. Tasks, data items, and relationships (represented by *data-flow edges*) between the two of them form the *data flow*. An example block-oriented cooking workflow for a sandwich recipe is illustrated in Fig. 1.

2.1 Semantic Workflows and Semantic Workflow Similarity

To support retrieval and adaptation of workflows, the individual workflow elements are annotated with ontological information, thus leading to a *semantic workflow* [2]. CookingCAKE uses a taxonomy of ingredients to define the semantics of data items and a taxonomy of preparation steps to define the semantics of tasks. These taxonomies are employed for the similarity assessment between tasks and data items. An example ingredient taxonomy is given in Fig. 2. A taxonomy is ordered by terms that are either a generalization or a specialization of a specific other term within the taxonomy, i.e., an inner node represents a generalized term that stands for the set of most specific terms below it. For example, the generalized term $vegeterian$ stands for the set $\{potatoes, rice, noodles\}$. Further on in the paper we use inner nodes in generalized workflows to represent that an arbitrary ingredient from the set of its specializations can be chosen.

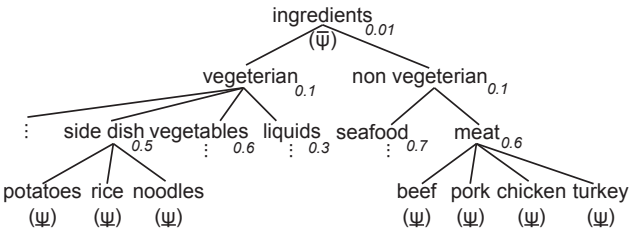

Fig. 2. Example of an ingredient taxonomy

In our previous work, we developed a semantic similarity measure for workflows that enables the similarity assessment of a case workflow W_c w.r.t. a query workflow W_q [2], i.e. $sim(W_q, W_c)$. Each query workflow element $x_q \in W_q$ is mapped by the function $m : W_q \rightarrow W_c$ to an element of the case workflow $x_c \in W_c$, i.e., $x_c = m(x_q)$. The mapping is used to estimate the similarity between the two workflow elements utilizing the taxonomy, i.e., $sim(x_q, x_c)$. The similarity of preparation steps or ingredients reflects the closeness in the taxonomy and further regards the level of the taxonomic elements. In general, the similarity is defined by the attached similarity value of the least common anchestor, e.g., $sim(beef, pork) = 0.6$. If a more general query element such as "meat" is compared with a specific element below it, such as "pork", the similarity value is 1. This ensures that if the query asks for a recipe containing meat, any recipe workflow from the case base containing any kind of meat is considered highly similar. All the similarity values of the mappings are then aggregated to estimate an overall workflow similarity.

2.2 Querying Semantic Workflows

In order to guide the retrieval and adaptation of workflows a query is defined by the user. CookingCAKE uses POQL (Query Language for Process-Oriented Case-Based Reasoning) [16] to capture desired and undesired ingredients or preparation steps of a cooking workflow as a query q. The definition of preparation steps is useful as certain tools might not be available or their usage is desired (e.g. oven). Let $q_d = \{x_1, \ldots, x_n\}$ be a set of desired ingredients or preparation steps and $q_u = \{y_1, \ldots, y_n\}$ be a set of undesired ingredients or preparation steps. A query q is then defined as $(x_1 \wedge \ldots \wedge x_2) \wedge \neg y_1 \wedge \ldots \wedge \neg y_n$. POQL also enables to capture generalized terms, i.e., if a vegeterian dish is desired, this can be defined by $\neg meat$. The query q is then used to guide retrieval, i.e., to search for a workflow which at best does not contain any undesired element and contains all desired elements. Based on the query q the unmatched elements can be identified, enabling estimating the elements to be deleted or added to the retrieved workflow during the subsequent adaptation stage. The similarity between the query and a workflow W is defined as the similarity between the desired ingredients and the workflow W and the number of undesired ingredients not contained in W according to the semantic similarity measure [2] in relation to the size of the query:

$$sim(q, W) = \frac{\sum_{x \in q_d} sim(x, m(x)) + |\{y \in q_u | sim(y, m(y)) \neq 1\}|}{|q_d| + |q_u|} \quad (1)$$

Hence, please note that similar desired ingredients or preparation steps increase the similarity while similar undesired ingredients or preparation steps do not reduce the similarity between the POQL query and the workflow.

In general, POQL is even more expressive and can, for example, capture time restrictions on preparation steps or that a certain ingredient should or should not be processed in a particular manner (e.g. do or do not bake vegetables). However, for the sake of simplicity we assume a set of desired and undesired ingredients or preparation steps only in the following sections.

3 Adaptation Approaches

This section summarizes the used adaptation approaches within the CookingCAKE framework.

3.1 Adaptation by Generalization and Specialization of Workflows

A generalized workflow [14] is a workflow containing generalized terms from a taxonomy (see Sec. 2.1), each of them representing multiple specialized ingredients or preparation steps. Thus, the generalized workflow represents a set of specialized workflows. Figure 3 illustrates an example for a generalization of the example workflow given in Fig. 1. Here, any preparation step that chops the cheese and any sort of meat could potentially be used. Such generalized workflows can be learned by comparing similar workflows from the case base. A workflow is generalized by generalizing terms if similar workflows from the case base contain several specializations of this generalized

term. It is assumed that if similar workflows contain the terms $\{beef, chicken, pork\}$, for example, these workflows can be generalized to contain any kind of $meat$. Likewise $makesmall$ represents all possible cooking steps reducing ingredients to small pieces.

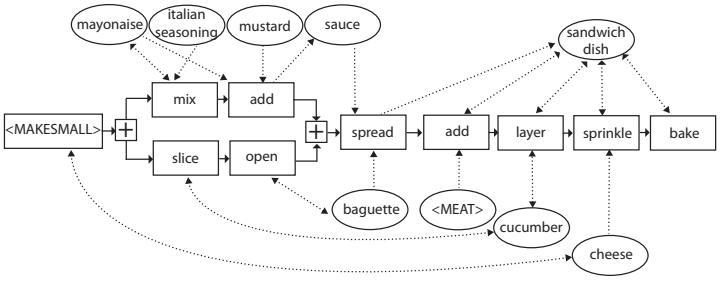

Fig. 3. Example of a generalized workflow

Adaptation is supported by specializing a workflow according to the POQL query q. Lets assume the generalized workflow contains the term $meat$ and the query defines that $beef$ is desired, the generalized element can be specialized according to $beef$. Thus, specialization enables adapting a workflow according to the POQL query.

3.2 Compositional Adaptation by Workflow Streams

The idea of compositional adaptation by workflow streams [13] is that each workflow can be decomposed into meaningful sub-components or snippets [7]. A sandwich workflow, for example, prepares the sauce and the toppings in order to produce the entire sandwich dish. These sub-components represented as partial workflows are referred to as *workflow streams*. Workflow streams can be identified by collecting all data-flow connected tasks[1] until a new data item such as sandwich sauce is created. An example for a workflow stream for the example workflow (see Fig. 1) is given in Figure 4 describing how to place toppings on the sandwich. To compute the adaptation knowledge, all workflow streams that can be found in the workflows within the case base are extracted.

The basic idea for compositional adaptation is, to adapt a workflow by using the workflow streams of other workflows that produce the same data item in a different manner, e.g., with other tasks or data. In the sandwich domain, for example, toppings, sauces, or preparation steps can be replaced. However, only workflow streams are substitutable if they produce the same data and consume identical data nodes. This ensures that replacing an arbitrary stream does not violate the semantic correctness of the workflow.

[1] If a task consumes a data item produced by another one, both tasks are dataflow-connected.

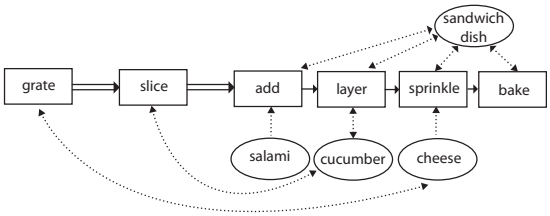

Fig. 4. Example of a workflow stream

3.3 Transformational Adaptation by Workflow Adaptation Operators

The workflow adaptation operators [15] are specified by two workflow sub-graphs called streamlets, one representing a workflow fraction to be deleted and one representing a workflow fraction to be added. Such operators can be learned from the case base by comparing two workflows and employ the difference between the two workflows in order to generate workflow adaptation operators. The example adaptation operator in Fig. 5 describes that mayonnaise can be replaced by tomatoes. This also enforces that tasks have to be changed as well, because the combine task also has to be exchanged for a chop task.

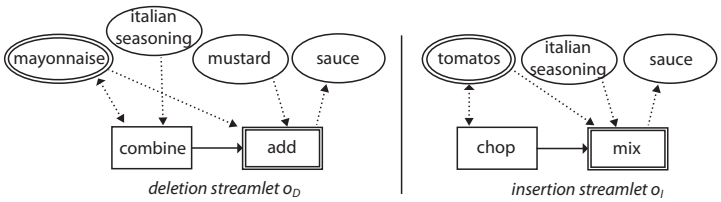

Fig. 5. Example of a workflow adaptation operator

The basic idea for operational adaptation is that chains of adaptation operators are applied $W \xrightarrow{o_1} W_1 \xrightarrow{o_2} \ldots \xrightarrow{o_n} W_n$ to the retrieved workflow W, thereby transforming the workflow W to an adapted workflow W_n. This process can be considered a search process towards an optimal solution w.r.t. the query. Hence, streamlets are removed, inserted, or replaced to transform the workflow according to the query.

4 CookingCAKE Framework

We now present the CookingCAKE framework which automatically generates adaptation knowledge using various adaptation approaches applied in POCBR (see Sect. 4.1). Based on this knowledge workflow adaptation is supported regarding a POQL query defining the requirements and resources on the workflow adaptation (see Sect. 4.2).

4.1 Generation of adaptation knowledge

As the acquisition of adaptation knowledge is an instance of the traditional knowledge acquisition bottleneck [5], CookingCAKE automatically generates adaptation knowledge based on the workflows contained in the case base (see Fig. 6). First, the case base and thus each workflow is generalized applying the method described in section 3.1. From this generalized case base further adaptation knowledge, i.e., workflow streams and adaptation rules (see Sect. 3), is automatically generated. As the adaptation knowledge is acquired based on the generalized case base, the adaptation knowledge itself is also generalized. This increases the adaptability for the entire adaptation procedure.

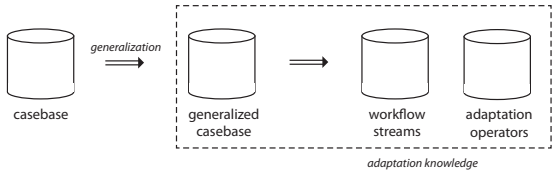

Fig. 6. Generation of adaptation knowledge

The generated adaptation knowledge can then be used to adapt a workflow whenever a query occurs. Further details on this procedure are explained in the next section.

4.2 Workflow adaptation

Whenever a POQL query occurs CookingCAKE searches for the workflow that best matches the given query within the generalized case base (see Fig. 7). However, it may happen that not all resources or requirements defined in the query are fulfilled by this workflow. Thus, workflow adaptation is required. For this purpose the workflow adaptation approaches presented in Sect. 3 are subsequently applied, still regarding the defined query. After this procedure, the adapted workflow still has to be specialized according to the query if it contains generalized elements. Therefore, CookingCAKE uses the specialization method presented in Sect. 3.1.

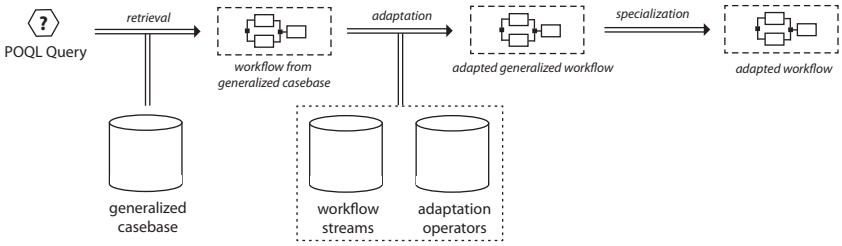

Fig. 7. Workflow adaptation

In order to ensure scalability of the presented approach for large case bases or large sets of adaptation knowledge, CookingCAKE supports a cluster-based retrieval [12] for workflows as well as for adaptation knowledge.

5 CCC Sandwich Challenge

In order to address the sandwich challenge a case base of 61 sandwich recipe workflows was created. The workflows were manually modelled based on sandwich recipes found on WikiTaaable[2] and further Internet sources. To enable similarity computations between the workflows, a modified version of the ingredient and cooking step ontology provided by WikiTaaable was employed. More precisely, multiple inheritance was resolved as CookingCAKE so far is only able to handle taxonomies (single inheritance). Further, the generalized terms of the taxonomies have been manually annotated with similarity values (see Sect. 2.1).

A running demo of CookingCAKE for the sandwich challenge is available under `http://cookingCAKE.wi2.uni-trier.de`[3] (see Fig. 8). The query of CookingCAKE contains desired and undesired ingredients as well as desired and undesired preparation

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

[3] Please note that CookingCAKE is still under improvement until the CCC'15

Fig. 8. Cooking Cake interface

steps. An example query (http://cookingCAKE.wi2.uni-trier.de?d=cherry%20tomato|salmon&u=cheese), generates a salmon and cherry tomato recipe without using any kind of cheese. Please note, that CookingCAKE does not necessarily fulfill the given query, it rather tries to fulfill the query as much as possible but does not execute any adaptations if no adaptation knowledge is present in order to remain the quality of the sandwich recipe. Consequently, if e.g. edam cheese is desired among other ingredients possibly a recipe with gouda cheese is returned if that is more suitable concerning the other desired desired ingredients. Further, undesired ingredients might be contained or a desired ingredient might not be contained, if that seems to be inappropriate according to the remaining ingredients given in the query. In general, cooking steps are adapted, if particular changed ingredients may require a different preparation of the particular dish.

After the definition of a query, CookingCAKE searches for the workflow in the case base that already best matches the given query based on the similarity value (see Sect. 2.2). If the query can not be fulfilled, adaptation is required. In this case, the entire adaptation procedure presented in Sect. 4.2 is used to adapt the sandwich recipe according to a query.

As a result, CookingCAKE can also print a detailed XML-File describing the used original case based recipe as well as the adapted recipe according to the query. Further, information is provided on which ingredients are removed from and added to the original workflow during adaptation.

Additionally, CookingCAKE also provides a textual view of the solution (see Fig. 9). For this purpose, the workflows are translated into a textual representation. Hence, the block-oriented workflow structure is reduced to a single sequence. Based on this, the required ingredients and the sequence of preparation steps (including the information on which ingredients are required in every preparation step) are generated. Further, the workflow itself is also illustrated in the process view.

CookingCAKE also features a name generator for the generated recipes. It accesses the taxonomy of ingredients and combines several terms of sub-taxonomies contained as ingredients in the workflow to assign a name to a recipe.

Based on the 61 recipes stored in CookingCAKE, generalization and specialization enable to generate more than $9 \cdot 10^{21}$ recipes. Further adaptations are supported by 197 workflow streams found and 7870 operators (1306 replace, 3903 insert, 2661 delete) generated. As the streams and operators are also generalized (see Sect. 4.2) adaptability is further increased. Hence, CookingCAKE provides a capable tool for the adaptation of sandwich recipes.

6 Conclusions and Future Work

We presented CookingCAKE, a framework for the adaptation of cooking recipes represented as workflows integrating and combining various adaptation approaches applied in Process-Oriented Case-Based Reasoning (POCBR). The available case base of cooking workflows is analyzed to generate adaptation knowledge which is used to adapt a recipe regarding a given query for the preparation of a dish.

In future work, we will investigate and integrate additional adaptation approaches for workflows such as the abstraction of workflows containing abstract tasks (e.g., pre-

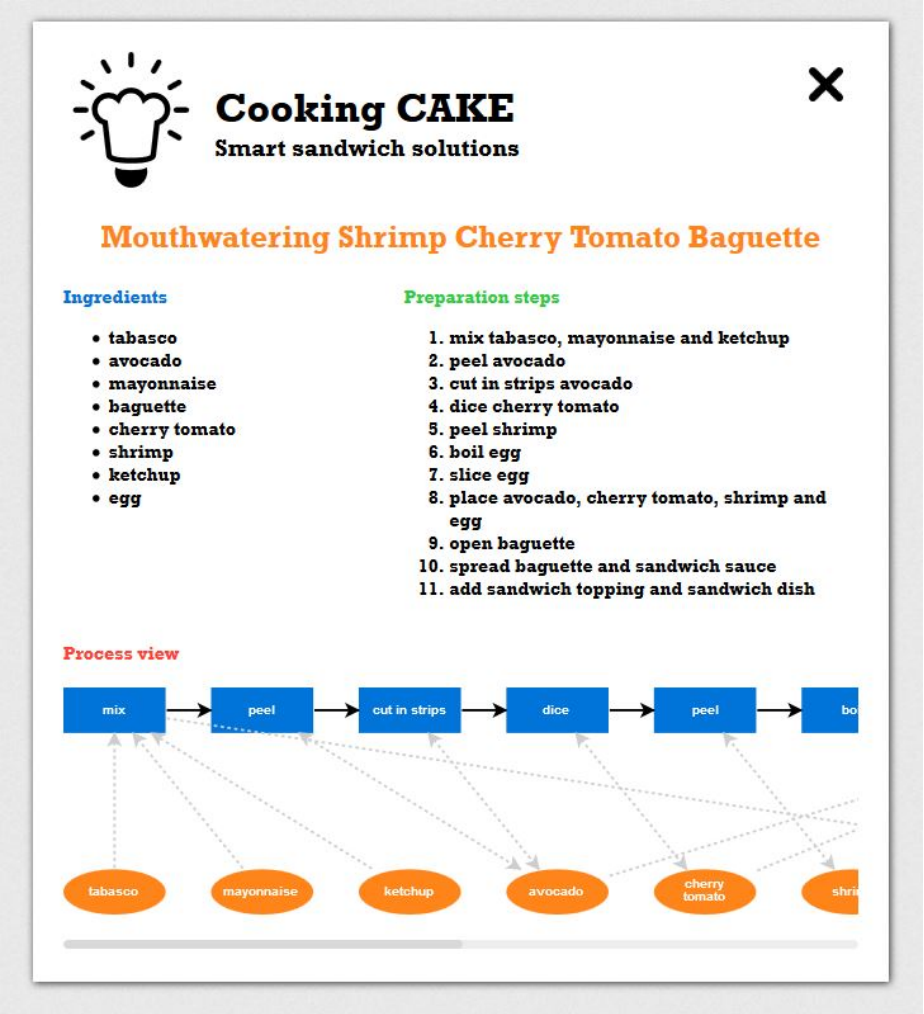

Fig. 9. Example recipe generated by CookingCAKE

pare sauce, place toppings on sandwich). Further, we will integrate the case-based adaptation approach of Minor et al. [10] in our framework. Moreover, CookingCAKE will be extended to be able to handle more knowledge-intensive ontologies, e.g., ontologies with multiple inheritance. Future work will also comprise the retrieval of adaptable cases [19], i.e., we will investigate the adaptability of the workflows within the case base as the workflow that best matches the given query is not necessarily the workflow that can be at best adapted to the resources and requirements given. Consequently, a better workflow as starting point for the adaptation can be chosen. Moreover, the retainment of adaptation knowledge[6] will be addressed by gathering user feedback on the adapted cooking recipes. This is important, as the quality of automatically learned

adaptation knowledge can not always be ensured. Thus, the quality of workflow adaptation is improved and the growth of adaptation knowledge can be controlled. Finally, CookingCAKE will be extended by interactive adaptation [1,8,20]. This supports the search of a suitable query by involving user interaction during adaptation which assist the user to create more individual cooking recipes.

Acknowledgements. This work was funded by the German Research Foundation (DFG), project number BE 1373/3-1.

References

1. Aha, D.W., Muñoz-Avila, H.: Introduction: Interactive case-based reasoning. Applied Intelligence 14(1), 7–8 (2001)
2. Bergmann, R., Gil, Y.: Similarity assessment and efficient retrieval of semantic workflows. Inf. Syst. 40, 115–127 (Mar 2014)
3. Dadam, P., Reichert, M., Rinderle-Ma, S., Göser, K., Kreher, U., Jurisch, M.: Von ADEPT zur AristaFlow BPM Suite-Eine Vision wird Realität:" Correctness by Construction" und flexible, robuste Ausführung von Unternehmensprozessen (2009)
4. Dufour-Lussier, V., Lieber, J., Nauer, E., Toussaint, Y.: Text adaptation using formal concept analysis. In: Bichindaritz, I., Montani, S. (eds.) Case-Based Reasoning. Research and Development, LNCS, vol. 6176, pp. 96–110. Springer (2010)
5. Hanney, K., Keane, M.: The adaptation knowledge bottleneck: How to ease it by learning from cases. In: David Leake, E.P. (ed.) Case-Based Reasoning Research and Development 1997, pp. 179–192. LNAI 1266, Springer (1997)
6. Jalali, V., Leake, D.: On retention of adaptation rules. In: Lamontagne, L., Plaza, E. (eds.) Case-Based Reasoning Research and Development, Lecture Notes in Computer Science, vol. 8765, pp. 200–214. Springer International Publishing (2014)
7. Kolodner, J.L. (ed.): Proceedings Case-Based Reasoning Workshop. Morgan Kaufmann Publishers, San Mateo, California (1988)
8. Leake, D.B., Wilson, D.C.: Combining CBR with interactive knowledge acquisition, manipulation and reuse. In: Case-Based Reasoning Research and Development, pp. 203–217. Springer (1999)
9. 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), 215–240 (2005)
10. 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 (2010)
11. Minor, M., Montani, S., Recio-Garca, J.A.: Process-oriented case-based reasoning. Information Systems 40(0), 103 – 105 (2014)
12. 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), IOS Press (2014)
13. Müller, G., Bergmann, R.: Workflow Streams: A Means for Compositional Adaptation in Process-Oriented Case-Based Reasoning. In: Proceedings of ICCBR 2014. Cork, Ireland (2014)
14. Müller, G., Bergmann, R.: Generalization of Workflows in Process-Oriented Case-Based Reasoning. In: 28th International FLAIRS Conference. AAAI, Hollywood (Florida), USA (2015)

15. Müller, G., Bergmann, R.: Learning and Applying Adaptation Operators in Process-Oriented Case-Based Reasoning. In: Proceedings of ICCBR 2015. Frankfurt, Germany (2015)
16. 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. COER Workshop Proceedings, to appear (2015)
17. Reichert, M.: Dynamische Ablaufänderungen in Workflow-Management-Systemen (2000)
18. Schumacher, P., Minor, M., Walter, K., Bergmann, R.: Extraction of procedural knowledge from the web. In: Workshop Proceedings: WWW'12. Lyon, France (2012)
19. Smyth, B., Keane, M.: Retrieving adaptable cases. In: Wess, S., Althoff, K.D., Richter, M. (eds.) Topics in Case-Based Reasoning, Lecture Notes in Computer Science, vol. 837, pp. 209–220. Springer Berlin Heidelberg (1994)
20. Weber, B., Wild, W., Breu, R.: CBRFlow: Enabling adaptive workflow management through conversational case-based reasoning. In: Advances in Case-Based Reasoning, pp. 434–448. Springer (2004)
21. Wilke, W., Bergmann, R.: Techniques and knowledge used for adaptation during case-based problem solving. In: Tasks and Methods in Applied Artificial Intelligence, pp. 497–506. Springer (1998)