Complexity-Aware Generation of Workflows by Process-Oriented Case-Based Reasoning

Gilbert Müller and Ralph Bergmann

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

Abstract. One of the biggest challenges in business process management is the creation of appropriate and efficient workflows. This asks for intelligent, knowledge-based systems that assist domain experts in this endeavor. In this paper we investigate workflow creation by applying Process-Oriented Case-Based Reasoning (POCBR). We introduce POCBR and describe how it can be applied to the experience-based generation of workflows by retrieval and adaptation of available best-practice workflow models. While existing approaches have already demonstrated their feasibility in principle, the generated workflows are not optimized with respect to complexity requirements. However, there is a high interest in workflows with a low complexity, e.g., to ensure the appropriate enactment as well as the understandability of the workflow. The main contribution of this paper is thus a novel approach to consider the workflow complexity during the workflow generation. Therefore, a complexity measure for workflows is proposed and integrated into the retrieval and adaptation process. An experimental evaluation with real cooking recipes clearly demonstrates the benefits of the described approach.

Keywords: case-based reasoning, process-oriented case-based reasoning, workflow complexity, workflow adaptation

1 Introduction

Business process management is a well-established discipline that deals with the identification, modeling, analysis, improvement, and implementation of business processes [1]. 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 to a set of procedural rules" [39]. In the recent years, the use of workflows has significantly expanded from the original domain of business processes towards new areas such as e-science [36], information integration [14], private activities [12], and even cooking [28]. One of the biggest challenges in the application of workflows today arises from the fact that workflows must be constructed or adapted more frequently [1,11,32]. 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.

In our research we address this problem by the application of Process-Oriented Case-Based Reasoning (POCBR) [25], which deals with the integration of Case-Based Reasoning (CBR) with process-oriented research areas like workflow management. POCBR aims at providing experience-based support for the automatic extraction [10], generation [20], execution [4], monitoring, and optimization [35,26] of workflows. We use POCBR to support the construction of workflows by reuse of already available workflows. Workflows are retrieved [6] and subsequently adapted [27,30,29] to new purposes and circumstances. More precisely, a case base (or repository) of successful workflows reflecting best-practices in a domain is the core of such 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. Using adaptation methods from CBR, workflows can be automatically adapted to better match the user's requirements.

While existing approaches have already demonstrated their feasibility in principle, the generated workflows are not optimized with respect to complexity requirements. However, complex workflows are more difficult to understand and to maintain, which may also result in a higher error-proneness of the workflow model [9]. Furthermore, the enactment of workflow models must not exceed the skills of the workflow participants or given time restrictions. In this paper, we address this issue by proposing a novel approach for the complexity-aware generation of workflows. Therefore, we introduce a measure for workflow complexity that can be used to asses created workflows automatically. Then, we adjust the POCBR methods for retrieval and adaptation such that this measure is considered during workflow creation. The proposed approach is fully implemented and demonstrated in the domain of cooking using workflows describing real cooking recipes. The complexity-aware creation of cooking workflows thus leads to new recipes which are easy to prepare.

The remainder of this paper is organized as follows: The next section presents the foundations of POCBR and our approaches for retrieval and adaptation. Then, we introduce a complexity measure for workflows and explain how the POCBR approaches can be adjusted to consider this measure. Finally, we present an experimental evaluation of the described approach.

2 Process-Oriented Case-Based Reasoning

Case-Based Reasoning is a problem solving paradigm built upon a rule of thumb suggesting that "similar problems tend to have similar solutions" [16,2,34]. The core of every case-based problem solver is a case base, which is a collection of memorized experience, called cases. The R^4-CBR cycle proposed by Aamodt and Plaza [2] consists of the four CBR phases retrieve, reuse, revise, and retain, which are performed sequentially when a new problem (also called new case or query) must be solved [22].

In POCBR a case is usually a workflow or process description expressing procedural experiential knowledge. Most basically, POCBR aims at generating new workflows or processes by retrieval and reuse of existing ones. The overall architecture the POCBR

approach we follow is illustrated in Fig. 1. First, a case base (or repository) of semantic workflows is constructed by selecting appropriate best-practice workflows from existing sources. The workflows in this case base can be reused, i.e., for a particular problem situation a suitable process represented as workflow can be suggested. This is primarily achieved by retrieving the best matching workflow from the repository. If required, the workflow is automatically adapted according to the requirements and restriction in the particular scenario. In the following of this section, we will summarize our related research in the field of POCBR.

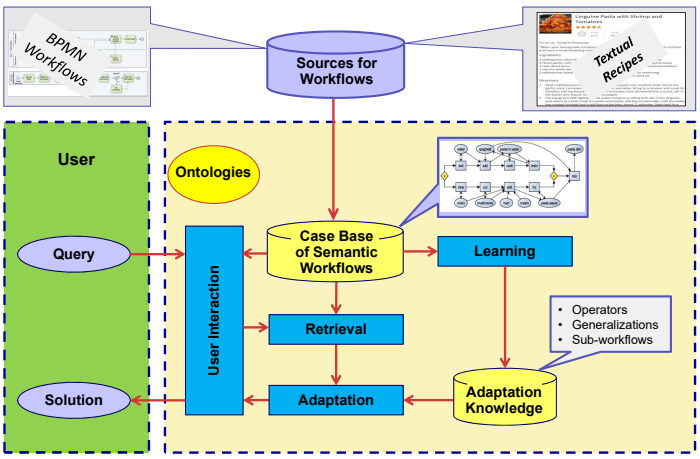

Fig. 1. POCBR Architecture

2.1 Semantic Workflows

In order to formalize procedural experience, we employed *semantic workflows* [6] as case representation. Broadly speaking, a workflow consists 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 (LOOP). In addition, tasks exchange certain *data items*, which can also be of physical matter. Tasks, data items, and relationships between the two of them form the *dataflow*. For the example application domain of cooking, a workflow describes the preparation steps required and ingredients used in order to prepare a particular dish. Here, the tasks represent the cooking steps and the data items refer to the ingredients being processed.

We consider workflows represented as a graph $W = (N, E)$ consisting of nodes $N = N^T \cup N^D \cup N^C$ and edges $E = E^C \cup E^D$. Nodes represent *tasks* N^T, *data nodes* N^D, or *control-flow nodes* N^C. The execution order of tasks is defined by *control-flow edges* $E^C \subseteq (N^T \cup N^C) \times (N^T \cup N^C)$ and the consumption or production of data nodes is specified by *dataflow edges* $E^D \subseteq (N^T \times N^D) \cup (N^D \times N^T)$. Furthermore, we enforce that each task $t \in N^T$ consumes ($\exists d \in N^D : (d, t) \in E^D$) and produces (i.e., $\exists d \in N^D : (t, d) \in E^D$) at least one data node. An example workflow graph for a sandwich recipe is illustrated in Fig. 2.

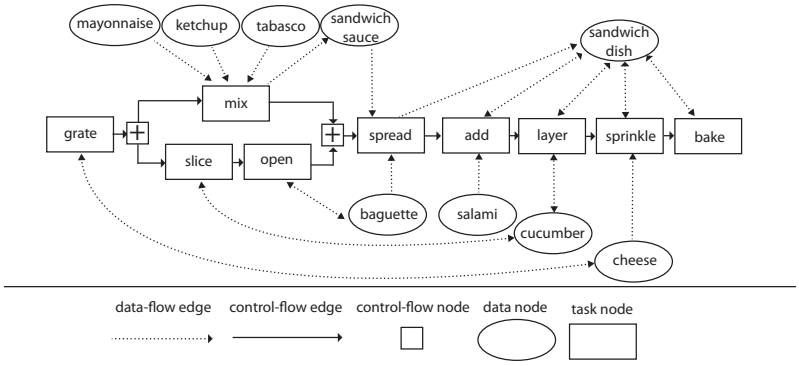

Fig. 2. Example of a block-oriented cooking workflow

To support retrieval and adaptation of workflows, the individual workflow elements are annotated with ontological information resulting in a *semantic workflow* [6]. We use a taxonomy of data items and a taxonomy of tasks to define the relevant aspects of their semantics and in particular as a means for similarity assessment. A taxonomy organizes the involved terms in a generalization/specialization hierarchy. In particular, an inner node represents a generalized term that stands for the set of more specific terms below it. An example data item taxonomy in the cooking domain is given in Figure 3 organizing cooking ingredients. For example, the generalized term $vegetarian$ stands for the set $\{potatoes, rice, noodles, \ldots\}$. Inner nodes in generalized workflows represent that an arbitrary ingredient from the set of its specializations can be chosen.

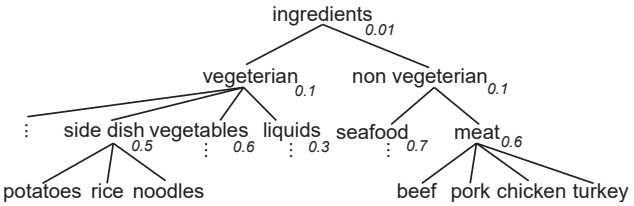

Fig. 3. Example of an ingredient taxonomy

2.2 Similarity-based Workflow Retrieval

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 [6], i.e., $sim(W_c, W_q)$. 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 tasks or data items 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 ancestor, e.g., $sim(beef, pork) = 0.6$ (see Fig. 3). 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 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.

To capture user requirements for a workflow to be generated, we employ POQL (Query Language for Process-Oriented Case-Based Reasoning) [31]. It allows to represent a query q consisting of desired and undesired data items and/or tasks of the workflow to be constructed. Let $q_d = \{x_1, \ldots, x_n\}$ be a set of desired data items or tasks and $q_u = \{y_1, \ldots, y_n\}$ be a set of undesired data items or tasks. A query q is defined as $(x_1 \wedge \ldots \wedge x_2) \wedge \neg y_1 \wedge \ldots \wedge \neg y_n$. POQL also enables the specification of generalized terms, i.e., if a vegetarian 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 contains all desired elements but no undesired element. Based on the query q the not matching elements can be identified, enabling to determine the elements to be deleted or added to the retrieved workflow during the subsequent adaptation stage. The query fulfillment for a query q and a workflow W is defined as the similarity between the desired tasks/data items and the workflow W and the number of undesired tasks/data items not contained in W according to the semantic similarity measure in relation to the size of the query:

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

Consequently, similar desired data items or tasks increase the query fulfillment, while matching undesired data items or tasks reduce the query fulfillment between the POQL query and the workflow.

2.3 Automatic Workflow Adaptation

We aim at supporting the users in situations in which the best matching workflow from the case base does not sufficiently fulfill the query. This requires that the workflow is automatically adapted according to the restrictions and requirements specified in the query, i.e., workflow elements or fragments are added or deleted according to the particular needs.

For that purpose, we developed several domain-independent workflow adaptation methods which we now briefly describe (for a more detailed description of these methods we refer to our previous work [30,27,29,28]). Since such adaptation methods usually require a significant amount of domain-specific adaptation knowledge, we additionally developed new methods that allow to automatically learn the required adaptation knowledge from the workflow repository. Hence, we distinguish between a learning phase of adaptation knowledge and a problem solving phase in which for a given query the best matching workflow is adapted such that it matches the particular problem scenario at best (see Fig. 1). The developed adaptation methods can mostly be classified into *transformational adaptation*, *compositional adaptation* and *adaptation by generalization* [22].

Transformational Adaptation The *operator-based adaptation* [30] is a *transformational adaptation* method in which the individual transformation steps are performed by so called *workflow adaptation operators*. They are denoted in a STRIPS-like manner. An operator consists of two workflow sub-graphs we call *streamlets*: a DELETE-streamlet specifies a workflow fragment to be deleted from the workflow and an ADD-streamlet represents a workflow fragment to be added to the workflow. The overall adaptation is implemented as a search process that aims at incrementally modifying the workflow with the goal to increase the query fulfillment (see formula 1). The required 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.

Compositional Adaptation The developed method for *compositional adaptation* is based on the idea that each workflow can be decomposed into meaningful sub-workflows called *workflow streams* [27]. 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. Such workflow streams can be automatically discovered from the workflow repository in the learning phase. Workflow streams represent valuable adaptation knowledge, which are used as "chunks" that can be used as replacement during compositional adaptation. More precisely, 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 syntactic correctness of the workflow. Compositional adaptation is also implemented as a search process that aims to increase query fulfillment, but it replaces larger portions of a workflow than transformational adaptation.

Adaptation by Generalization and Specialization of Workflows Finally, *generalization and specialization* was investigated as a third adaptation approach [29]. A *generalized workflow* is structurally identical to the base workflow but the semantic descriptions

of task and data items are generalized. We generalize a workflow by considering a set of similar workflows as training samples and employ the taxonomy as generalization hierarchy from which generalized semantic descriptions are selected. The computed generalized cases are then stored as a generalized workflow repository. During problem solving, adaptation is performed by specializing a previously generalized workflow in a manner such that query fulfillment is maximized.

Integration of Adaptation Methods The three approaches can be integrated to form a hybrid adaptation approach that combines the three adaptation capabilities [28]. This integration involves the actual adaptation process as well as the learning phase.

First of all, during the learning phase, adaptation operators and workflow streams can be learned not only from the available specific workflows, but also from workflows resulting from generalization. Thus, we first apply generalization to the workflows in the case base and then we learn adaptation operators and workflow streams from the generalized workflows. The learned knowledge can then be used when adapting generalized workflows retrieved from the generalized case base.

The adaptation process itself then uses the three adaptation methods in combination (see Fig. 4). First, similarity-based retrieval selects the best matching workflow from the generalized case base of semantic workflows. Then, compositional adaptation is applied. Thereby, entire sub-workflows (e.g., the preparation of the *sandwich sauce*) are replaced by matching learned sub-workflows (e.g., other *sauces*) from other workflows. Next, adaptation is performed by applying adaptation operators, which result in additional modifications of the workflows (e.g., the ingredient *tomato* is replaced by *mushrooms*). Finally, the workflows are specialized (if necessary) by replacing single generalized data items or tasks by means of the specified taxonomy (e.g. the generalized ingredient *meat* is replaced by *chicken*).

In all approaches, the adaptation of the workflow is performed by chaining several adaptation steps $w \overset{\alpha_1}{\rightarrow} w_1 \overset{\alpha_2}{\rightarrow} \ldots \overset{\alpha_n}{\rightarrow} w_n = w'$, which iteratively transforms the retrieved workflow w towards an adapted workflow w'. This process solves an optimization problem aiming at maximizing the query fulfillment (as specified in formula 1). This optimization process is implemented as a heuristic search procedure with the goal to achieve an adapted workflow with the highest query fulfillment possible. In the domain of

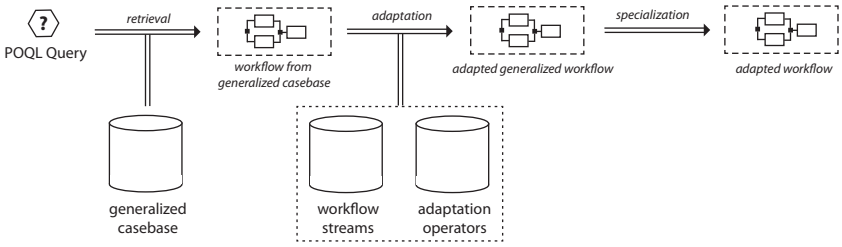

Fig. 4. Integration of adaptation approaches

cooking, this integrated adaptation process aims at adding missing desired ingredients-/preparation steps and at removing undesired contained ingredients/preparation steps specified in the query.

3 Complexity Assessment of Workflows

We now summarize related work on workflow complexity and then introduce a novel criterion for the retrieval and adaptation process that considers the complexity of workflows in addition to the query fulfillment. Thus, retrieval and adaptation become complexity-aware by optimizing the constructed workflow also in this regard.

3.1 Business Process Complexity

In business process literature, several assessment approaches exist that aim at measuring process quality. Existing quality models [21] propose quality attributes, criteria, and predicates. They relate to the efficacy and efficiency of business processes and measure the quality of process design, implementation, and enactment. In addition, *quality metrics* have been proposed for business process models [37,23,13] as a means for the systematic and automated assessment of certain quality attributes. The discussions of process model quality also involves complexity as an important issue. Several authors discuss process complexity from the perspective of pragmatic quality referring to the understandability of the process model [33,24]. Various complexity measures for process models have been proposed in the literature. A study performed by Latva-Koivisto [18], for example, suggests the use of graph complexity measures (e.g, *coefficient network complexity* or a *complexity index*). Furthermore, various approaches assess the complexity of business processes based on established measures in software development [8,7,19]. For example, Cardoso et al. [9] present complexity measures that include the number of activities or control-flow elements, consider the complexity induced by control-flow nodes, or the complexity resulting from the dataflow. As an alternative approach, Vanderfeesten et al. [38] introduce the cross-connectivity metric considering the connectivity strength between process elements. For process models based on petri nets (referred to as workflow nets) Lassen and Van der Aalst [17] presented several complexity metrics. However, despite the large number of different complexity measures, no standard approach for assessing the complexity of workflows currently exists.

3.2 Complexity-Aware Query Fulfillment

In order to consider complexity during the construction process, a new complexity-aware criterion is required. In this work, we consider the complexity of a workflow W as a generic function $complexity(W) \rightarrow [0, 1]$ that assigns a high value to workflows with a high complexity and a lower value to less complex workflows.

$$QF_{\text{complexity}}(q, W) = \alpha \cdot QF(q, W) + (1 - \alpha) \cdot (1 - complexity(W)) \qquad (2)$$

Based on this complexity function, we define a new complexity-aware query fulfilment measure $QF_{\text{complexity}}(q, W) \rightarrow [0, 1]$ (see Eq. 2). Both criteria are weighted by a parameter $\alpha \in [0, 1]$. The complexity-aware query fulfilment measure replaces the original measure specified in formula 1. It is used during the retrieval process to select the best matching workflow from the case base and during the hybrid adaptation process. Thus, it aims at optimizing the constructed workflow also with regard to complexity. Please note that this results in a multi-objective optimization problem and thus the adaptation may not be able to maximize the query fulfillment and to reduce the complexity of the workflow at the same time.

3.3 Complexity Measure

Due to the various approaches and perspectives to measure workflow complexity presented in the literature, we assume that the complexity of a workflow is not determined by a single feature, but is composed of several criteria. We introduce a new complexity function $complexity(W) \rightarrow [0, 1]$ that covers five different indicators for determining the complexity of a workflow (see Table 1).

Table 1. Complexity criteria

critera description	criteria measure
number of data nodes	$\frac{\lvert N^D \rvert}{max\{\lvert N_1^D \rvert,\ldots,\lvert N_n^D \rvert\}}$
number of control-flow elements	$\frac{\lvert N^T \cup N^C \rvert}{max\{\lvert N_1^T \cup N_1^C \rvert,\ldots,\lvert N_n^T \cup N_n^C \rvert\}}$
complexity of dataflow	$1 - \frac{2 \cdot \lvert N^T \rvert}{\lvert E^D \rvert}$
complexity of tasks	$\frac{\sum_{t \in N^T} taskComplexity(t)}{\lvert N^T \rvert}$
lead time	$\frac{leadTime(W)}{max\{leadTime(W_1),\ldots,leadTime(W_n)\}}$

The first two criteria measure basic complexity properties, i.e., the *number of control-flow elements* (task and control-flow nodes) as well as the *number of data nodes* in the workflow $W = (N, E)$. Both measures are normalized by the highest amount of control-flow elements or data nodes contained in a workflow from the case base. Consequently, workflows with more data nodes or more control-flow elements are assumed to be more complex. Furthermore, the *complexity of tasks* as well as the *complexity of dataflow* represent two additional complexity criteria. The complexity measure for the *dataflow* considers the average amount of data nodes consumed and produced by the tasks, which assigns a high complexity value to those workflows in which the

tasks N^T consume and produce a large amount of data nodes[1] E^D. For computing the *complexity of tasks*, which determine the required skill level for executing the task [21], each task t in the taxonomy is annotated by an estimated task complexity value $taskComplexity(t) \in [0, 1]$. The criterion is defined as the average complexity of the tasks in the workflow W. Finally, the total time to execute a particular workflow (also referred to as *lead time* [15]) is also considered as an indicator for the workflow complexity. Therefor, each task t in the taxonomy is annotated by an approximated throughput time $throughputTime(t) \in \mathbb{N}$. The throughput time [15] measures the execution time of a task t as the elapsed time between finishing the previous task and finishing the particular task t. The *lead time* for a workflow W is then heuristically measured by aggregating the throughput times of the tasks. We denote $maxPath(W) \subseteq N^T$ as those tasks that are part of the longest sequence path with regard to highest total throughput time from start to end node. The overall lead time is then computed by adding up the single throughput times of the tasks in $maxPath(W)$, i.e., $leadTime(W) = \sum_{t \in maxPath(W)} throughputTime(t)$. To assess the corresponding complexity, this value is normalized in relation to the workflows from the repository as defined in Table 1. The overall complexity measure $complexity(W)$ of a workflow W is then defined as the arithmetic mean of these five complexity criteria.

3.4 Complexity in the Cooking Domain

As our evaluation is performed in the cooking domain, we will now briefly describe the implications of the domain-independent complexity measure described in the previous section on cooking recipes. In the cooking domain, the complexity of workflows basically determines the difficulty of preparation. Due to several reasons, amateur chefs may search for easy-to-prepare cooking recipes with a low complexity. Thus, the complexity-aware generation of cooking workflows is a highly relevant application field.

We assume that also in cooking the complexity is composed by various criteria that are mostly reflected by the introduced complexity measure. Thus, we measure the *number of preparation steps* (number of control-flow nodes) and *number of ingredients* (number of data nodes) as two basic complexity criteria. Furthermore, the complexity of dataflow determines the *complexity of the preparation*, assuming that recipes in which each preparation step consumes and produces a high amount of ingredients is more complex. The application of task complexity in the cooking domain measures the *complexity of preparation steps*. For instance, the preparation step *blanche* is more complex than the preparation step *mix*. The lead time denotes the duration time of preparation, which is another indicator for the preparation complexity. The preparation step *mix*, for example, is considered as short, while *baking* results in a longer preparation time.

[1] Please note that each task in a workflow consumes and produces at least one data node, respectively.

4 Experimental Evaluation

We implemented the described approach for complexity-aware workflow generation by extending our generic POCBR system CAKE [5]. Using CAKE, we also developed an application in the cooking domain, called CookingCAKE [28], which uses a workflow repository of 61 sandwich recipe workflows manually modelled from various Internet sources (e.g., sandwich recipes on WikiTaaable [3][2]). The resulting workflows are purely sequential, thus no control-flow nodes occur. The employed taxonomies of preparation steps and ingredients are based on the WikiTaaable ontology and were manually annotated with similarity, preparation time, and task complexity values.

A running prototype of the complexity-aware workflow generation in CookingCAKE is available under `http://cookingCAKE.wi2.uni-trier.de/complexity`. The query of CookingCAKE involves desired and undesired ingredients as well as desired and undesired preparation steps. An example query could ask to generate a *salmon* and *cherry tomato* recipe without using any kind of *cheese*. CookingCAKE then selects the best matching workflow from the repository and subsequently adapts it according to the novel criterion $QF_{complexity}(q, W)$. Thus, the system tries to maximize the query fulfilment on the one hand and on the other hand aims at reducing the complexity of the workflow to generate an appropriate easy-to-prepare recipe for an amateur chef.

To evaluate the complexity-aware approach for workflow construction in the cooking domain, we performed several leave-one-out experiments. We generated 61 queries automatically as follows: for each workflow W in the repository, a corresponding query was constructed by selecting the most similar workflow W' from the repository and by determining the difference between the two workflows. The constructed query considers workflow elements as desired that are only contained in the workflow W and considers the elements only contained in workflow W' as undesired. At most 4 randomly selected ingredients and 2 preparation steps are determined as desired or undesired respectively. For each query we executed the described POCBR approach to generate an appropriate workflow, while the workflow from which the query was derived was temporarily removed from the repository. We performed this experiment with the standard approach, applying only the query fulfillment criterion as well as with the complexity-aware approach. For the complexity-aware recipe construction we chose the parameter $\alpha = 0.5$ to consider the query fulfillment and the complexity in equal shares. For both approaches, we measured the query fulfillment, the complexity, and the combined complexity-aware criterion of the retrieved as well as of the adapted workflow.

Table 2. Evaluation results: average values over all queries

	query fulfillment	complexity	combined	computation time
standard retrieval	0.83	0.43	0.70	1.15 s
standard adaption	0.92	0.48	0.72	18.73 s
complexity-aware retrieval	0.75	0.28	0.74	1.42 s
complexity-aware adaption	0.87	0.29	0.79	9.49 s

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

The evaluation results illustrated in Table 2 clearly demonstrate several benefits of the presented approach. First of all, the experiments confirm our previous experimental evaluations of the adaptation methods [27,30,29] by showing that they lead to a significant increase in query fulfillment (from 0.83 to 0.92 for the standard approach and from 0.75 to 0.87 for the complexity-aware approach). For the domain of cooking this means that cooking recipes are generated that are close to the users requirements. Second, the impact of the proposed complexity-aware query fulfillment approach is clearly visible. During retrieval, less complex workflows are selected as starting point for adaptation. In both approaches adaptation tends to increase the workflow complexity in favor of query fulfillment. However, altogether with the complexity-aware approach, the complexity of the generated workflow is significantly reduced (-40%), while the query fulfillment itself is only slightly decreased (-5%). Furthermore, the overall computation time is significantly decreased[3].

5 Conclusions and Future Work

This paper presents a new approach to consider workflow complexity during the automatic generation of workflows in a case-based manner. The complexity measure is composed of several criteria including the number of tasks and data items, the complexity of the tasks and the dataflow as well as the lead time. We demonstrated the benefits of the presented approach in an experimental study in the domain of cooking based on real cooking recipes and ontologies.

In future work, the complexity assessment will be extended and evaluated by considering various other complexity and quality measures (see Sect. 3.1). So far, we demonstrated our methods primarily in the domain of cooking. Thus, we further aim at broadening the experimental basis by exploring existing workflow and business process model repository collections. In particular, we will investigate the field of scientific text mining workflows in more detail.

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

References

1. van der Aalst, W.M.: Business process management: a comprehensive survey. ISRN Software Engineering 2013 (2013)
2. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Communications 7(1), 39–59 (1994)
3. Badra, F., Bendaoud, R., Bentebibel, R., Champin, P., Cojan, J., Cordier, A., Desprès, S., Jean-Daubias, S., Lieber, J., Meilender, T., Mille, A., Nauer, E., Napoli, A., Toussaint, Y.: TAAABLE: text mining, ontology engineering, and hierarchical classification for textual case-based cooking. In: Schaaf, M. (ed.) ECCBR 2008, Workshop Proceedings. pp. 219–228 (2008)

[3] The adaptation time depends on the size of the workflow, which is usually smaller, if a less complex workflow is retrieved.

4. 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. pp. 519–533. ECCBR'06, Springer, Berlin, Heidelberg (2006)
5. 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), http://doi.acm.org/10.1145/2660398.2663771
6. Bergmann, R., Gil, Y.: Similarity Assessment and Efficient Retrieval of Semantic Workfows. Information Systems 40, 115–127 (2014)
7. Cardoso, J.: About the data-flow complexity of web processes. In: 6th International Workshop on Business Process Modeling, Development, and Support: Business Processes and Support Systems: Design for Flexibility. pp. 67–74 (2005)
8. Cardoso, J.S.: Business process control-flow complexity: Metric, evaluation, and validation. Int. J. Web Service Res. 5(2), 49–76 (2008), http://dx.doi.org/10.4018/jwsr.2008040103
9. Cardoso, J.S., Mendling, J., Neumann, G., Reijers, H.A.: A discourse on complexity of process models. In: Eder, J., Dustdar, S. (eds.) Business Process Management Workshops, BPM 2006 International Workshops, BPD, BPI, ENEI, GPWW, DPM, semantics4ws, Vienna, Austria, September 4-7, 2006, Proceedings. LNCS, vol. 4103, pp. 117–128. Springer (2006), http://dx.doi.org/10.1007/11837862_13
10. 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)
11. Fleischmann, A., Schmidt, W., Stary, C., Augl, M.: Agiles Prozessmanagement mittels Subjektorientierung. HMD Praxis der Wirtschaftsinformatik 50(2), 64–76 (2013)
12. Görg, S., Bergmann, R.: Social workflows - Vision and potential study. Information Systems 50, 1–19 (2015), only preprint of this article available for download
13. Heravizadeh, M., Mendling, J., Rosemann, M.: Dimensions of Business Processes Quality (QoBP). In: van der Aalst, W., Mylopoulos, J., Sadeh, N.M., Shaw, M.J., Szyperski, C., Ardagna, D., Mecella, M., Yang, J. (eds.) Business Process Management Workshops, vol. 17, pp. 80–91. Springer (2009)
14. Hung, P., Chiu, D.: Developing workflow-based information integration (WII) with exception support in a web services environment. In: System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on. p. 10–pp (2004)
15. Jansen-Vullers, M., Loosschilder, M., Kleingeld, P., Reijers, H.: Performance measures to evaluate the impact of best practices. In: Proceedings of Workshops and Doctoral Consortium of CAiSE'07 (BPMDS workshop). vol. 1, pp. 359–368. Tapir Academic Press Trondheim (2007)
16. Kolodner, J.: Case Based Reasoning. Morgan Kaufmann (1993)
17. Lassen, K.B., van der Aalst, W.M.P.: Complexity metrics for workflow nets. Information & Software Technology 51(3), 610–626 (2009), http://dx.doi.org/10.1016/j.infsof.2008.08.005
18. Latva-Koivisto, A.M.: Finding a complexity measure for business process models. Helsinki University of Technology, Systems Analysis Laboratory (2001)
19. Laue, R., Gruhn, V.: Complexity metrics for business process models. In: BIS. LNI, vol. 85, pp. 1–12. GI (2006)
20. 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, pp. 203–217. ICCBR '99, Springer, London, UK (1999)
21. Lohrmann, M., Reichert, M.: Understanding Business Process Quality. In: Business Process Management - Theory and Applications, pp. 41–73. Springer, Berlin Heidelberg (2013)

22. 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)
23. Mendling, J.: Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness, LNBIP, vol. 6. Springer (2008)
24. Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven process modeling guidelines (7PMG). Information & Software Technology 52(2), 127–136 (2010), http://dx.doi.org/10.1016/j.infsof.2009.08.004
25. Minor, M., Montani, S., Recio-Garca, J.A.: Process-oriented case-based reasoning. Information Systems 40(0), 103 – 105 (2014)
26. Montani, S., Leonardi, G.: Retrieval and clustering for supporting business process adjustment and analysis. Inf. Syst. 40, 128–141 (2014)
27. Müller, G., Bergmann, R.: Workflow Streams: A Means for Compositional Adaptation in Process-Oriented Case-Based Reasoning. In: Case-Based Reasoning. Research and Development, ICCBR 2014. pp. 315–329. LNCS 8766, Springer (2014)
28. Müller, G., Bergmann, R.: CookingCAKE: A framework for the adaptation of cooking recipes represented as workflows. In: Kendall-Morwick, J. (ed.) Workshop Proceedings from ICCBR 2015, Frankfurt, Germany, September 28-30, 2015. CEUR Workshop Proceedings, vol. 1520, pp. 221–232. CEUR-WS.org (2015)
29. Müller, G., Bergmann, R.: Generalization of Workflows in Process-Oriented Case-Based Reasoning. In: Proceedings of FLAIRS 2015. pp. 391–396. AAAI Press, Hollywood (Florida), USA (2015)
30. Müller, G., Bergmann, R.: Learning and Applying Adaptation Operators in Process-Oriented Case-Based Reasoning. In: Case-Based Reasoning. Research and Development, ICCBR 2015. pp. 259–274. LNCS 9343, Springer (2015)
31. Müller, G., Bergmann, R.: POQL: A new query language for process-oriented case-based reasoning. In: Bergmann, R., Görg, S., Müller, G. (eds.) Proceedings of the LWA 2015 Workshops: KDML, FGWM, IR, and FGDB, Trier, Germany, October 7-9, 2015. CEUR Workshop Proceedings, vol. 1458, pp. 247–255. CEUR-WS.org (2015)
32. Reichert, M., Weber, B.: Enabling flexibility in process-aware information systems: challenges, methods, technologies. Springer Science & Business Media (2012)
33. Reijers, H.A., Mendling, J., Recker, J.: Business process quality management. In: vom Brocke, J., Rosemann, M. (eds.) Handbook on Business Process Management 1, Introduction, Methods, and Information Systems, 2nd Ed., pp. 167–185. International Handbooks on Information Systems, Springer (2015), http://dx.doi.org/10.1007/978-3-642-45100-3_8
34. Richter, M.M., Weber, R.O.: Case-Based Reasoning - A Textbook. Springer (2013)
35. Sauer, T., Maximini, K.: Using Workflow Context for Automated Enactment State Tracking. In: Minor, M. (ed.) Workshop Proceedings: ECCBR 2006, Workshop: Case-based Reasoning and Context Awareness. pp. 300–314. Universität Trier (2006)
36. Taylor, I.J., Deelman, E., Gannon, D.B.: Workflows for e-Science. Springer (2007)
37. Vanderfeesten, I., Cardoso, J., Mendling, J., Reijers, H.A., Van der Aalst, W.: Quality metrics for business process models. BPM and Workflow handbook 144, 179–190 (2007)
38. Vanderfeesten, I.T.P., Reijers, H.A., Mendling, J., van der Aalst, W.M.P., Cardoso, J.S.: On a quest for good process models: The cross-connectivity metric. In: Bellahsene, Z., Léonard, M. (eds.) Advanced Information Systems Engineering, CAiSE 2008, Montpellier, France, 2008, Proceedings. LNCS, vol. 5074, pp. 480–494. Springer (2008), http://dx.doi.org/10.1007/978-3-540-69534-9_36
39. Workflow Management Coalition: Workflow management coalition glossary & terminology (1999)