

# Compositional Adaptation of Cooking Recipes using Workflow Streams

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.** This paper presents first steps toward a compositional adaptation approach of cooking recipes represented as cooking workflows. The available case base of cooking workflows is analyzed and each workflow is decomposed into meaningful subcomponents, called workflow streams. During adaptation, deficiencies in the retrieved cooking workflows are compensated by replacing fragments of the retrieved cooking workflow by appropriate workflow streams. This approach regards a broad range of restrictions and resources, the user may define for the preparation of a dish.

**Keywords:** recipe adaptation, compositional adaptation, workflows

## 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 where a broad range of resources must be regarded, e.g., the availability of cooking tools, ingredients, time, and human capacities. Further, restrictions such as human cooking skills or consumer diets have to be considered.

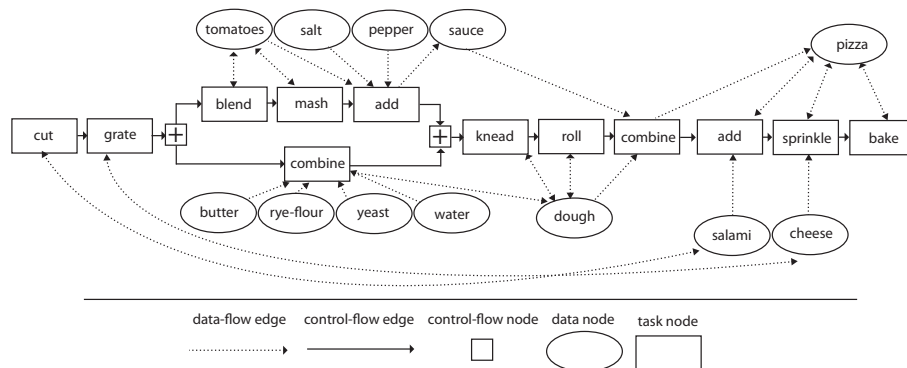
As direct processing of textual recipes is almost not feasible, they are usually transformed to structured cases, e.g., workflows [8]. However, as there exists only little research so far, workflow adaptation is an important field in Process-Oriented Case-Based Reasoning (POCBR).

Existing methods for adaptation in CBR can be roughly classified into transformational, compositional, and generative adaptation [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. [4]. Compositional adaptation usually means that several cases are used during adaptation, incorporating transformational or generative adaptation methods involving adaptation knowledge. Instead, we investigate a pure compositional adaptation approach.

Dufour-Lussier et al. [3] already presented an approach for compositional adaptation, which differs as in their work cooking processes are represented as trees and user preferences concerning ingredients are regarded. In contrast, we present first ideas toward an approach that considers a broad range of criteria to adapt a workflow regarding the restrictions and resources mentioned before (e.g., ingredients, time, and human cooking skills). The available case base of workflows is analyzed and each case is decomposed into meaningful subcomponents, called *workflow streams* [5]. During adaptation, deficiencies in the retrieved recipes are compensated by replacing fragments of the retrieved case by appropriate workflow streams. It is enforced that recipes are represented as block-oriented workflows, which ensures the semantic and syntactic correctness of the workflow streams and the adapted workflow. Hence, the next section introduces block-oriented cooking workflows. In Section 3, the workflow streams are defined as a prerequisite for the compositional adaptation method described in Section 4. Finally, we wrap-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 [8] (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 [7,2], 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 the execution order. This forms the *control-flow*. Tasks, data items, and relationships (represented



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

by *data-flow edges*) between the two of them form the *data flow*. An example block-oriented cooking workflow for a pizza recipe is illustrated in Figure 1.

## 2.1 Partial Workflows

A partial workflow  $W'$  of a cooking workflow  $W$  is defined for a subset of tasks and contains the subset of data nodes that are linked to any task in  $W'$  as well as those control-flow blocks that construct a workflow w.r.t. the block-oriented workflow structure. Control-flow blocks containing no tasks or other control-flow blocks, for example, are not part of the partial workflow.  $W'$  contains a subset of control-flow and data-flow edges connecting any two elements of  $W'$ . Furthermore  $W'$  is supplemented by a set of additional control-flow edges that retain the execution order of the control-flow. Figure 2 illustrates a partial workflow  $W'$  of the workflow  $W$  given in Figure 1. One additional edge is required in this example (depicted by the double-line arrow) as “grate” and “add” are not linked in  $W$ .

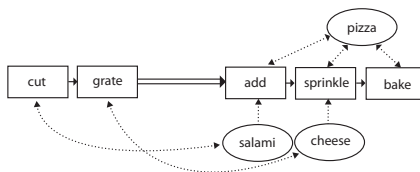


Fig. 2. Example of a partial workflow  $W'$

## 2.2 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* [1]. Hence, based on a manually constructed *cooking ontology*, additional information such as required skills, cooking tools, or time can be annotated to preparation steps. Further, this ontology may also contain information about ingredients (e.g., dietary information). All preparation steps and ingredients are organized in a taxonomy, which enables the assessment of similarity among them. In our previous work, we developed a semantic similarity measure for workflows that enables the similarity assessment of a case workflow w.r.t. a query workflow [1]. The similarity of preparation steps or ingredients reflects the closeness in the taxonomy and further regards the level of the taxonomic elements. In particular, 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.

The similarity measure performs a kind of inexact subgraph matching, optimizing the overall similarity between the matched workflow elements. It is used during case retrieval in order to find workflows which best match a certain query. During adaptation, the same similarity based retrieval method is used to identify reusable workflow streams, as we will show in Section 4.

### 3 Workflow Streams

We automatically partition a workflow  $W$  into a set of partial workflows representing subprocesses of the entire preparation process. These partial workflows are referred to as workflow streams. We define that two tasks  $t_1, t_2$  are data-flow connected if  $t_1$  is executed prior to  $t_2$  and  $t_1$  produces an ingredient consumed by  $t_2$ . A workflow stream contains a set of transitively data-flow connected tasks and each task of  $W$  is contained exactly in one stream of  $W$ . The set of transitively data-flow connected task is split by preparation steps producing new ingredients (referred to as *creator tasks*, see  $\odot$  in Fig. 3), which mark the last preparation step of a workflow stream. Hence, streams containing a creator task represent subprocesses that create new ingredients. In the workflow illustrated in Figure 3, these streams create the dough, the sauce, or combine the dough and sauce to create the pizza (see stream 2,3,4). The task “mash tomatoes”, for example, does not belong to the workflow stream 4 producing the pizza, although it is transitively data-flow connected, as it is separated through a data-flow connected creator task that produces the pizza sauce consumed by stream 4 (see “combine” step of stream 4). Further, the remaining set of transitively data-flow connected tasks is also used to construct streams processing ingredients (streams without a creator task). As an example, see stream 1 in Fig. 3 for putting the toppings on the pizza. The extracted workflow streams are consistent as they are block-oriented workflows. Furthermore, due to the data-flow connectedness, the streams maintain their “semantic correctness” as they represent meaningful connected subcomponents of the original workflow.

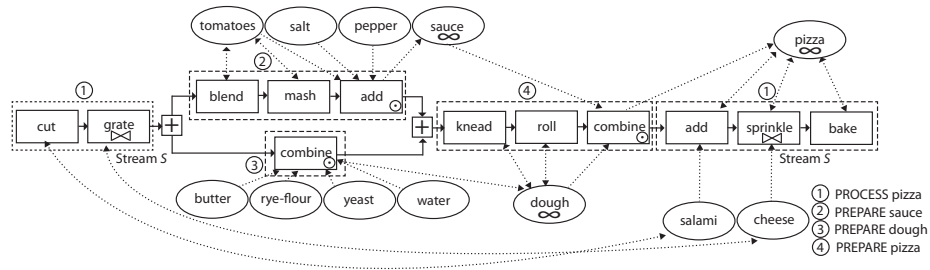


Fig. 3. Workflow  $W$  and workflow streams

The basic idea for compositional adaptation is, to adapt a workflow by using the workflow streams of other workflows that represent similar processes, e.g.,

with other ingredients or preparation steps. We make use of specific ingredients, referred to as *anchors* of a stream, which connect two streams via the data-flow. Thus, anchors of a stream are ingredients of this stream that are either produced (pre anchor) or consumed (post anchor) by a preparation step of another stream of the same workflow. The anchors of all streams are marked with  $\infty$  symbols in Figure 3, i.e., sauce, dough, and pizza.

## 4 Compositional Adaptation using Workflow Streams

In the following, we assume that each workflow in the case base has been automatically decomposed into workflow streams according to Section 3. These streams are linked to each workflow in the case base and stored in a separate workflow stream repository. We now present an example scenario of how to replace a workflow stream by another workflow stream. Based on this we are going to describe how the retrieved workflow can be adapted according to the users' preferences.

### 4.1 Change Request

Following the retrieval, a change request is defined by specifying the changes that should be made to the recipe. This change request is either manually defined or automatically generated by the deficiency of the query and the retrieved recipe such as desired/undesired ingredients or time restrictions given in the query that are not met by the retrieved workflow. Accessing the cooking ontology during adaptation also enables, e.g., to regard dietary restrictions as this information is annotated to ingredients. Furthermore, this change request can comprise, among other things, which cooking tools should be used or must not be used, number of available cooks, or the skill of the cooks.

### 4.2 Replacing Workflow Streams

Assuming workflow  $W$  given in Figure 3 should be adapted. To replace a stream  $\mathcal{S}$  with another stream  $\mathcal{S}'$ , the stream  $\mathcal{S}$  is first removed from the workflow, by constructing a partial workflow of  $W$  containing all tasks of  $W$  except of those contained in  $\mathcal{S}$ .

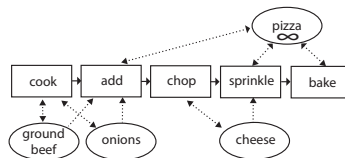
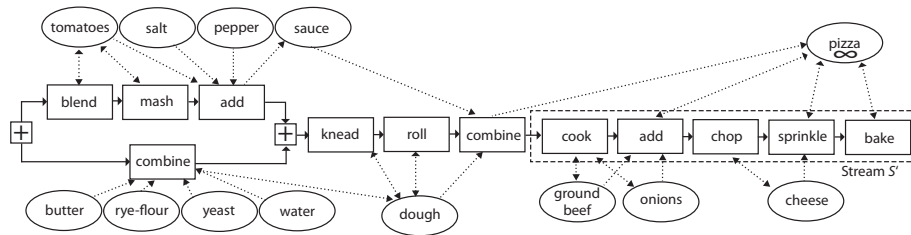


Fig. 4. Substitute stream  $\mathcal{S}'$

Then, the new stream  $\mathcal{S}'$  is inserted at the position of the last control-flow element (preparation step or control-flow block) of the workflow stream  $\mathcal{S}$  in  $W$ . This means that all edges, preparation steps, and ingredients (if not already present) of  $\mathcal{S}'$  are inserted into the workflow  $W$ . Then, the inserted stream  $\mathcal{S}'$  is connected with additional control-flow edges that link the first and/or last control-flow element to the workflow  $W$ . In the illustrated scenario, the stream  $\mathcal{S}'$  is inserted behind the last “combine” task (see Fig. 5).

Inserting the stream  $\mathcal{S}'$  may lead to a shift of preparation steps. In the given scenario, for example, the start node of the original workflow and the adapted workflow differ (compare Fig. 3 and Fig. 5). However, this does not violate the semantic correctness of the workflow, i.e., the ingredients are still being processed in the right order.



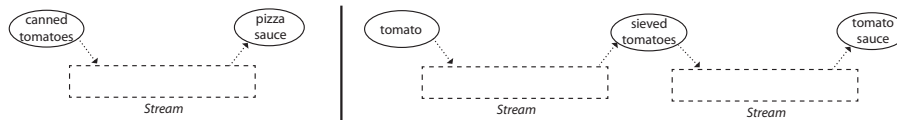
**Fig. 5.** Stream  $\mathcal{S}$  removed and stream  $\mathcal{S}'$  added to  $W$

### 4.3 Compositional Adaptation

During the automatic compositional adaptation, workflow streams that violate the requirements defined in the change request are replaced by more appropriate workflow streams. Furthermore, streams might be replaced if a new inserted stream may fulfill a requirement that is not met by any stream given in the workflow. This could also be context-dependent, i.e., instead of applying a change request to the entire workflow, a change has only be made to a specific stream.

Which workflow streams can replace a stream of the workflow in general, is determined by the usage of the anchor ingredients (see Sec. 3). The set of post anchors of the workflow streams must be identical or at least highly similar (e.g., pizza sauce and tomato sauce, see Fig. 6). All streams that fulfill this premise for a workflow stream are referred to as substitute candidates. To replace a stream, the substitute candidates are searched for a similar stream that in addition fulfills the given change request. Regarding the similarity between the stream to be replaced and the substitute candidate ensures that the performed adaptation only changes the workflow in a minimal way, just enough that the required changes are made. As opposed to this, disregarding the similarity enables to replace very different processes that produce fairly similar products (e.g., “tomato sauce”,

“sauce hollandaise”). Thus, the impact of the similarity between the streams during adaptation enables to configure this trade-off.



**Fig. 6.** Stream Chains

Inserting a new workflow stream could enforce the insertion of an additional stream as it might require further ingredients that have to be preprocessed during the dish preparation (see Fig. 6). Thus, a set of streams (referred to as *stream chain*) rather than a single stream is inserted. Hence, streams are iteratively searched for streams with post anchors that are identical to the current pre anchors until either the pre anchors are already contained in the workflow or the pre anchors are not contained as a post anchor of any other stream. In the latter case the pre anchors can be regarded as raw ingredients (as they are not produced by any preparation step). An example is illustrated in Figure 6. Assuming that the pizza sauce stream is going to be replaced by the tomato sauce stream, also the sieved tomatoes stream has to be added if only tomatoes are present. Consequently, other streams might have to be deleted as they become superfluous since their products are never used (e.g., when replacing the tomato sauce stream with the pizza sauce stream).

Furthermore, the compositional adaptation is able to parallelize specific workflow streams if there is more than one cook available as it is not required that all of them are executed in a sequential order. Making the dough and preparing the sauce, for example, can be done by two different cooks at the same time.

## 5 Conclusions and Future Work

We presented first steps toward a novel approach to compositional adaptation of cooking workflows by decomposing them into reusable workflow parts (workflow streams). We investigated how to identify and to replace workflow streams and developed an approach to adapt entire workflows regarding a broad range of restrictions and resources defined by the user. The proposed approach ensures that the adapted workflows are consistent (w.r.t. the block structure) and semantically correct (w.r.t. the use of the ingredients).

Currently, we achieved first satisfactory results for a limited approach replacing workflow streams having identical anchors, able to regard preparation steps and ingredients the user prefers or refuses [5]. In future work, we will extend this approach with the ideas presented in this paper, e.g., the stream chains and the change request. Furthermore, we will explore various ways to improve

the proposed method, e.g., by constructing different kinds of workflow streams. More precisely, other criteria have to be identified for how to separate a workflow into workflow streams with different sizes. This could be used to apply the presented adaptation approach multiple times with different workflow stream configurations (if necessary), starting from the largest workflow stream size configuration. This probably retains a good adaptation quality, as replacing larger workflow streams reduces adaptation errors. On the other hand, this will most likely enable the application of more adaptations, and thus increase the probability to fulfill the change request, as smaller workflow streams can be more easily adapted. However, the presented approach is limited, as it requires similarly structured workflows and workflow streams to be present and is restricted to block-oriented workflows. Further studies will also comprise the investigation of how the presented approach is related to case-based planning [6].

Moreover, we will use workflow streams to construct abstract workflows. These abstract workflows can be used as a skeleton, i.e. a cooking workflow containing abstract tasks (e.g., prepare dough, prepare sauce, place toppings on pizza), to create entire cooking workflows from scratch by replacing the abstract preparation tasks by appropriate workflow streams and thus by concrete tasks.

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

## References

1. Bergmann, R., Gil, Y.: Similarity assessment and efficient retrieval of semantic workflows. *Inf. Syst.* 40, 115–127 (Mar 2014)
2. 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)
3. Dufour-Lussier, V., Lieber, J., Nauert, 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)
4. 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)
5. 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)
6. Muñoz-Avila, H., Cox, M.T.: Case-based plan adaptation: An analysis and review. *Intelligent Systems, IEEE* 23(4), 75–81 (2008)
7. Reichert, M.: *Dynamische Ablaufänderungen in Workflow-Management-Systemen* (2000)
8. Schumacher, P., Minor, M., Walter, K., Bergmann, R.: Extraction of procedural knowledge from the web. In: *Workshop Proceedings: WWW’12*. Lyon, France (2012)
9. 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)