Computer Cooking Contest

Workshop at the
Twenty-Third International Conference on
Case-Based Reasoning
(ICCBR 2015)

Frankfurt, Germany
September 2015

Emmanuel Nauer and David C. Wilson (Editors)

Program Chairs

Emmanuel Nauer	Loria, Université de Lorraine, France
David C. Wilson	University of North Carolina at Charlotte, USA

Technical Chair

Emmanuelle Gaillard	Loria, Université de Lorraine, France

Program Committee

David Aha	Naval Research Laboratory, USA
Klaus-Dieter Althoff	University of Hildesheim, Germany
Ralph Bergmann	University of Trier, Germany
Isabelle Bichindaritz	State University of New York at Oswego, USA
Ichiro Ide	Nagoya University, Japan
Luc Lamontagne	Université Laval, Canada
David Leake	Indiana University, USA
Jean Lieber	LORIA, Université de Lorraine, France
Mirjam Minor	Goethe University, Germany
Santiago Ontañón	Drexel University, USA
Miltos Petridis	Brighton University, UK
Ralph Traphöner	empolis GmbH, Germany
Nicolas Valance	Groupe SEB, France

Preface

We are happy to present the contributions of four teams that have been accepted to the Computer Cooking Contest 2015. The Computer Cooking Contest (CCC) is an open competition. All individuals (e.g., students, professionals), research groups, and others are invited to submit software that creates recipes. The primary knowledge source is a database of basic recipes from which appropriate recipes can be selected, modified, or even combined. The queries to the system will include the desired and undesired ingredients. For most of the queries there is no single correct or best answer. That is, many different solutions are possible, depending on the creativity of the software. There is no restriction on the technology that may be used; all are welcome. The only restriction is that the given database of recipes must be used as a starting point.

The 8th Computer Cooking Contest will be held in conjunction with the 2015 International Conference on Case-Based Reasoning in Frankfurt, Germany. A web site with detailed information is online at: computercookingcontest.net. There are three challenges:

1. **Cocktail challenge on making real cocktails**
   In this challenge, the system should be able to suggest a tasty cocktail recipe that matches a user query including a set of desired ingredients from a limited set of available ingredients and avoiding unwanted ones (not necessarily from the limited set of ingredients). In addition, the system should adapt the ingredient quantities. Without information on the ingredient quantities, the original quantities will be used to prepare the cocktail.
   - *Evaluation criteria*: scientific quality, culinary quality
   - *Assessment procedure*: paper evaluation and comparison of the results of the systems on a same set of queries by the jury (cocktail jury prize) and public vote after tasting in real the recipes of all the system on a same query chosen by the jury (cocktail public prize).
   - *Material provided by the organizers (see resources section)*: the cocktail case base of 109 cocktail recipes, semantically annotated according to the Wikitaaable ontology; access to the WikiTaaable ontology (wikitaaable.loria.fr); specified basic set of ingredients available to prepare a cocktail.
2. **Sandwich challenge on making real sandwiches**
   In this challenge, the system should be able to suggest a tasty cold sandwich recipe that matches a user query including a set of desired ingredients and avoiding unwanted ones. In addition, the system should adapt the recipe preparation, at least the order in which ingredient will be put in the sandwich. Without information on the ingredient quantities, the original input procedure will be used to prepare the sandwich in real. The recipe will be interpreted by a chef, to correct some usual missing preparation step, for example, put the tomatoes without mentioning that the tomatoes must be sliced.
   - *Evaluation criteria*: scientific quality, culinary quality

- *Assessment procedure*: paper evaluation and comparison of the results of the systems on a same set of queries by the jury (sandwich jury prize), and public vote after tasting in real the recipes of all the system on a same query chosen by the jury (sandwich public prize).
- *Material provided by the organizers*: the sandwich case base - a set of 21 sandwich recipes, semantically annotated according to the Wikitaaable ontology; access to the WikiTaaable ontology (wikitaaable.loria.fr); an additional database containing 9507 sandwich recipes crawled from the web.

3. **Open challenge on adapting cooking recipes**
   In this challenge, you may propose whatever you want about the adaptation of cooking recipes, e.g. workflow adaptation, text adaptation, community-based adaptation, recipes combination, explanations, similarity computation, recipe personalized recommendation, etc. The evaluation will take into account the originality aspect and the scientific aspect of the work. A running system implementing the work is optional.
   - *Evaluation criteria*: scientific quality, originality, culinary quality
   - *Assessment procedure*: usual scientific review process; program committee vote

We would like to thank all contributors, reviewers, local organizers, the jury, and our sponsors: Empolis, INRIA Nancy Grand Est, and LORIA, who kindly provided financial support for the CCC. We are looking forward to try some excellent sandwiches and fascinating, novel cocktails in Frankfurt.

September 2015 — Emmanuel Nauer
Frankfurt — David C. Wilson

Improving Ingredient Substitution using Formal Concept Analysis and Adaptation of Ingredient Quantities with Mixed Linear Optimization

Emmanuelle Gaillard, Jean Lieber, and Emmanuel Nauer

Université de Lorraine, LORIA — 54506 Vandœuvre-lès-Nancy, France
CNRS — 54506 Vandœuvre-lès-Nancy, France
Inria — 54602 Villers-lès-Nancy, France
firstname.lastname@loria.fr

Abstract. This paper presents the participation of the TAAABLE team to the 2015 Computer Cooking Contest. The TAAABLE system addresses the *mixology* and the *sandwich* challenges. For the *mixology challenge*, the 2014 TAAABLE system was extended in two ways. First, a formal concept analysis approach is used to improve the ingredient substitution, which must take into account a limited set of available foods. Second, the adaptation of the ingredient quantities has also been improved in order to be more realistic with a real cooking setting. The adaptation of the ingredient quantities is based on a mixed linear optimization. The team also applied TAAABLE to the *sandwich challenge*.

Keywords: case-based reasoning, formal concept analysis, adaptation of ingredient quantities, mixed linear optimization.

1 Introduction

This paper presents the participation of the TAAABLE team to the *mixology* and to the *sandwich* challenges of the 2015 Computer Cooking Contest (CCC). The TAAABLE system is based on many methods and techniques in the area of knowledge representation, knowledge management and natural language processing [1]. Currently, it is built over TUUURBINE (`http://tuuurbine.loria.fr`), a generic case-based reasoning (CBR) system over RDFS [2] which allows reasoning over knowledge stored in a RDF store, as the one provided by the contest.

For this edition of the CCC, TAAABLE has been extended in order to improve the ingredient substitution procedure which must manage unavailable foods. An approach based on formal concept analysis (FCA) allows improving ingredient substitutions. Moreover, the adaptation of the ingredient quantities has also been improved in order to be more realistic with a real cooking setting. The adaptation of the ingredient quantities is based on mixed linear optimization. This adaptation takes into account the preference unit given in the source recipe and proposes quantities which are usual. For example, when the ingredient is a lemon, its quantity will take the form of a human easy understandable value (i.e. a quarter, a half, etc. instead of *54 g*, which corresponds to a half lemon).

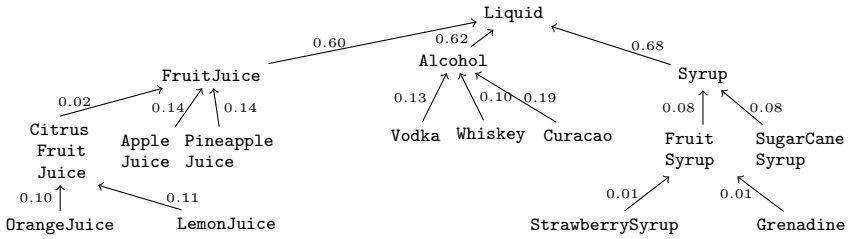

Fig. 1. The hierarchy forming the domain knowledge used in the running example with the generalization costs used as retrieval knowledge.

Section 2 introduces the core of the TAAABLE system. Section 3 details the new approaches developed specially for the mixology challenge. Section 4 explains the system submitted for the sandwich challenge.

2 The TAAABLE system

The challenges, proposed by the CCC since its first edition consists in proposing, according to a set of initial recipes, one or more recipes matching a user query composed of a set of wanted ingredients and a set of unwanted ingredients. The TAAABLE system addresses this issue through an instantiation of the generic CBR TUUURBINE system [3], which implements a generic CBR mechanism in which adaptation consists in retrieving similar cases and in replacing some features of these cases in order to adapt them as a solution to a query.

2.1 TUUURBINE founding principles

TUUURBINE is a generic CBR system over RDFS. The domain knowledge is represented by an RDFS base `DK` consisting of a set of triples of the form $\langle C \ \texttt{subClassOf} \ D\rangle$ where C and D are classes which belong to a same hierarchy (e.g, the food hierachy). Fig. 1 represents the domain knowledge for the running examples by a hierarchy whose edges $C \xrightarrow{x} D$ represent the triples $\langle C \ \texttt{subClassOf} \ D\rangle$. The retrieval knowledge is encoded by a cost function: $\texttt{cost}(\langle C \ \texttt{subClassOf} \ D\rangle) = x$ for an edge $C \xrightarrow{x} D$. This cost can be understood intuitively as the measure of "the generalization effort" from C to D. How this cost is computed is detailed in [1].

A TUUURBINE case `case` is described by a set of triples of the form $\langle URI_{\texttt{case}} \ \texttt{prop} \ \texttt{val}\rangle$, where $URI_{\texttt{case}}$ is the URI of `case`, `val` is either a resource representing a class of the ontology or a value and `prop` is an RDF property linking `case` to a hierarchy class or to the value. For simplification, in this paper, we represent a case by a conjunction of expressions only of the form `prop` : `val`. For example, the "Rainbow" recipe is represented by the following index `R`, which means that "Rainbow" is a cocktail recipe made from vodka, orange juice, grenadine and curacao (`ing` stands for *ingredient*).

$$\begin{aligned}\texttt{R} = \ & \texttt{dishType}:\texttt{CocktailDish} \\ & \wedge \texttt{ing}:\texttt{Vodka} \wedge \texttt{ing}:\texttt{OrangeJuice} \wedge \texttt{ing}:\texttt{Grenadine} \wedge \texttt{ing}:\texttt{Curacao}\end{aligned} \tag{1}$$

For instance, the first conjunct of this expression means that the triple $\langle URI_R$ `dishType CocktailDish`$\rangle$ belongs to the knowledge base.

2.2 TUUURBINE query

A TUUURBINE query is a conjunction of expressions of the form `sign prop : val` where `sign` $\in \{\epsilon, +, !, -\}$, `val` is a resource representing a class of the ontology and `prop` is an RDF property belonging to the set of properties used to represent cases. For example,

$$\begin{aligned} \texttt{Q} = {} & +\texttt{dishType} : \texttt{CocktailDish} \\ & \wedge \texttt{ing} : \texttt{Vodka} \wedge \texttt{ing} : \texttt{Grenadine} \wedge !\texttt{ing} : \texttt{Whiskey} \end{aligned} \tag{2}$$

is a query to search "a cocktail with vodka and grenadine syrup but without whiskey".

The signs ϵ (*empty sign*) and $+$ are "positive signs": they prefix features that the requested case must have. $+$ indicates that this feature must also occur in the source case whereas ϵ indicates that the source case may not have this feature, thus the adaptation phase has to make it appear in the final case.

The signs ! and $-$ are "negative signs": they prefix features that the requested case must not have. $-$ indicates that this feature must not occur in the source case whereas ! indicates that the source case may have this feature, and, if so, that the adaptation phase has to remove it.

2.3 TUUURBINE retrieval process

The retrieval process consists in searching for cases that best match the query. If an exact match exists, the corresponding cases are returned. For the query `Q` given in (2), the "Rainbow" recipe is retrieved without adaptation. Otherwise, the query is relaxed using a generalization function composed of one-step generalizations, which transforms `Q` (with a minimal cost) until at least one recipe of the case base matches $\Gamma(Q)$.

A one step-generalization is denoted by $\gamma = \texttt{prop} : \texttt{A} \rightsquigarrow \texttt{prop} : \texttt{B}$, where `A` and `B` are classes belonging to the same hierarchy with $\texttt{A} \sqsubseteq \texttt{B}$, and `prop` is a property used in the case definition. This one step-generalization can be applied only if `A` is prefixed by ϵ or ! in `Q`. If `A` is prefixed by !, thus `B` is necessarily the top class of the hierarchy. For example, the generalization of $!\texttt{ing} : \texttt{Rum}$ is $\epsilon\texttt{ing} : \texttt{Food}$, meaning that if rum is not wanted, it has to be replaced by some other food. Classes of the query prefixed by $+$ and $-$ cannot be generalized.

Each one-step generalization is associated with a cost denoted by $\texttt{cost}(A \rightsquigarrow B)$. The generalization Γ of `Q` is a composition of one-step generalizations $\gamma_1, \ldots \gamma_n$: $\Gamma = \gamma_n \circ \ldots \circ \gamma_1$, with $\texttt{cost}(\Gamma) = \sum_{i=1}^{n} \texttt{cost}(\gamma_i)$. For example, for:

$$\begin{aligned} \texttt{Q} = {} & +\texttt{dishType} : \texttt{CocktailDish} \\ & \wedge \texttt{ing} : \texttt{Vodka} \wedge \texttt{ing} : \texttt{PineappleJuice} \wedge \texttt{ing} : \texttt{Grenadine} \wedge !\texttt{ing} : \texttt{Whiskey} \end{aligned} \tag{3}$$

`PineappleJuice` is relaxed to `FruitJuice` according to the domain knowledge of Fig. 1. At this first step of generalization, $\Gamma(\texttt{Q}) =$

dishType : CocktailDish⊓ing : Vodka⊓ing : FruitJuice⊓!ing : Whiskey, which matches the recipe described in (1), indexed by OrangeJuice, a FruitJuice.

2.4 Tuuurbine adaptation process

When the initial query does not match existing cases, the cases retrieved after generalization have to be adapted. The adaptation consists of a specialization of the generalized query produced by the retrieval step. According to Γ(Q), to R, and to DK, the ingredient OrangeJuice is replaced with the ingredient PineappleJuice in R because FruitJuice of Γ(Q) subsumes both OrangeJuice and PineappleJuice. Tuuurbine implements also an adaptation based on rules where some ingredients are replaced with others in a given context [4]. For example, in cocktail recipes, replacing OrangeJuice and StrawberrySyrup with PineappleJuice and Grenadine is an example of an adaptation rule. This rule-based adaptation is directly integrated in the retrieval process by searching cases indexed by the substituted ingredients for a query about the replacing ingredients, for example by searching recipes containing OrangeJuice and StrawberrySyrup for a query about PineappleJuice and Grenadine.

2.5 Taaable as a Tuuurbine instantiation

The Taaable knowledge base is WikiTaaable (http://wikitaaable.loria.fr/), the knowledge base made available for this CCC edition. WikiTaaable is composed of the four classical knowledge containers: (1) the domain knowledge contains an ontology of the cooking domain which includes several hierarchies (about food, dish types, etc.), (2) the case base contains recipes described by their titles, the dish type they produce, the ingredients that are required, the preparation steps, etc., (3) the adaptation knowledge takes the form of adaptation rules as introduced previously, and (4) the retrieval knowledge, which is stored as cost values on subclass-of relations and adaptation rules.

In WikiTaaable, all the knowledge (cases, domain knowledge, costs, adaptation rules) is encoded in a triple store, because WikiTaaable uses Semantic Media Wiki, where semantic data is stored into a triple store. So, plugging Tuuurbine over the WikiTaaable triple store is quite easy because it requires only to configure Tuuurbine by giving the case base root URI, the ontology root URI and the set of properties on which reasoning may be applied.

3 Mixology challenge

The mixology challenge consists in retrieving a cocktail that matches a user query according to a set of available foods given by the CCC organizers (white rum, whiskey, vodka, orange juice, pineapple juice, sparkling water, coca-cola, beer grenadine syrup, lemon juice, mint leaves, lime, ice cube, brown sugar, salt, and pepper). Tuuurbine queries can express this kind of request using the ϵ and ! prefixes. Section 3.1 explains how the user query is transformed to take into account only the available foods, before being submitted to Tuuurbine. Two additionnal processes have been implemented to improve the Tuuurbine adaptation result. The first process searches, when some ingredients of the source

recipe are not available, the best way to replace them, or in some cases, to remove them (see Section 3.2). The second process uses REVISOR/CLC (see Section 3.4) to adapt quantities. A new formalization of the quantity adaptation problem is proposed to obtain more realistic quantity values, taking into account the type of unit given in the source case (see Section 3.4).

3.1 Query building

For the mixology challenge, where an answer must only contain the available food, the query may be built by adding to the initial user query the minimal set of classes of the food hierarchy that subsume the set of foods which are not available, each class being negatively prefixed by !. For example, let us assume that `OrangeJuice` and `PineappleJuice` are the only available fruit juices, that `Vodka` and `Whiskey` are the only available alcohols, that `SugarCaneSyrup` and `Grenadine` are the only available syrups, and that the user wants a cocktail recipe with `Vodka` but without `SugarCaneSyrup`. The initial user query will be $\texttt{Q} = +\texttt{dishType}:\texttt{CocktailDish} \wedge \epsilon\texttt{ing}:\texttt{Vodka} \wedge !\texttt{ing}:\texttt{SugarCane}$. According to Fig. 1, `LemonJuice`, `AppleJuice`, `Curacao`, and `StrawberrySyrup` will be added to this initial query with a ! for expressing that the result cannot contain one of these non available classes of food, which includes their descendant classes. The extended query `EQ` submitted to TUUURBINE will be:

$$\begin{aligned}\texttt{EQ} = \texttt{Q} &\wedge !\texttt{ing}:\texttt{LemonJuice} \wedge !\texttt{ing}:\texttt{AppleJuice} \\ &\wedge !\texttt{ing}:\texttt{StrawberrySyrup} \wedge !\texttt{ing}:\texttt{Curacao}\end{aligned}$$

For this example, TUUURBINE retrieves the "Rainbow" recipe with the adaptation "replace `Curacao` with `Food`", due to $!\texttt{ing}:\texttt{Curacao}$.

In order to replace `Curacao` by something more specific than `Food`, a new approach based on FCA is proposed.

3.2 Using FCA to search the best ingredient substitution

When ingredients of the source case must be replaced because these pieces of food are not available, we choose FCA to exploit ingredient combination in cocktail recipes in order to search which ingredient(s) is/are the most used with the ones already used in the recipe that must be adapted. FCA is a classification method allowing object grouping according to the properties they share [5]. FCA takes as input a *binary context*, i.e. a table in which objects are described by properties. Table 1 shows an example of binary context with 7 objects (which are cocktails), described by two kinds of properties: the ingredients they use, and some more generic ingredient classes: `_Alcohol`, the generic class of recipes with at least one alcohol, and `_Sugar`, the generic class of recipes with at least one sweet ingredient, like sugar or syrup. These generic classes are prefixed by `_` to be distinguished from the concrete ingredients. For example, the object `Screwdriver` has the properties `Vodka` and `Orange juice` (the ingredients used in this cocktail), and `_Alcohol`, because `Vodka` is an alcohol.

FCA produces *formal concepts* as output. A formal concept is a pair (I, E) where I is a set of properties, E is a set of objects, respectively called the *intent*

	_Alcohol	Vodka	White rum	Tequila	Cacha ca	Blue cucacao	Orange juice	Coca-cola	Lime	_Sugar	White sugar	Cane sugar syrup	Grenadine
Screwdriver	×	×					×						
Rainbow	×	×				×	×			×			×
Tequila sunrise	×			×			×			×			×
Ti'Punch	×		×						×	×		×	
Daiquiri	×		×						×	×		×	
Caipirinha	×				×				×	×	×		
Cuba libre	×		×					×	×				

Table 1. A binary context for cocktails, described by their ingredients and two generic food classes (`_Alcohol` and `_Sugar`).

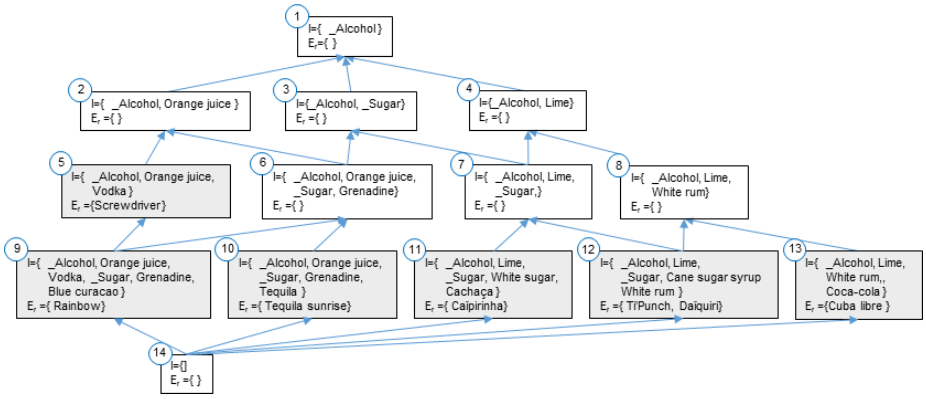

Fig. 2. Concept lattice organizing cocktails according to their ingredients.

and the *extent* of the formal concept, such that (1) I is the set of all properties shared by the objects of E and (2) E is the set of all objects sharing properties in I. The formal concepts can be ordered by extent inclusion, also called *specialisation* between concepts, into what is called a *concept lattice*. Fig. 2 illustrates the lattice corresponding to the binary context given in Table 1. On this figure, the extents E are given through a reduced form (noted E_r): the objects appear in the most specific concepts, the complete extent can be computed by the union of objects belonging to the subconcepts. So, the top concept (#1, in the figure) contains all the objects. In our example, its intent is `_Alcohol`, a property shared by all the objects. By contrast, the bottom concept is defined by the set of all properties. In our example, its extent is empty as none of the objects are described by all the properties.

To search a replacing ingredient in a given recipe or in a recipe according to pieces of food that will be kept, the idea is to exploit the lattice which captures concept similarities and organization. For example, concept #7, which intent is {`_Alcohol`, `Lime`, `_Sugar`}, allows an access to 3 cocktails containing at least one alcohol, at least one sugar, and lime. Adapting a cocktail can be based on the closeness between concepts. For example, when a replacing ingredient is searched

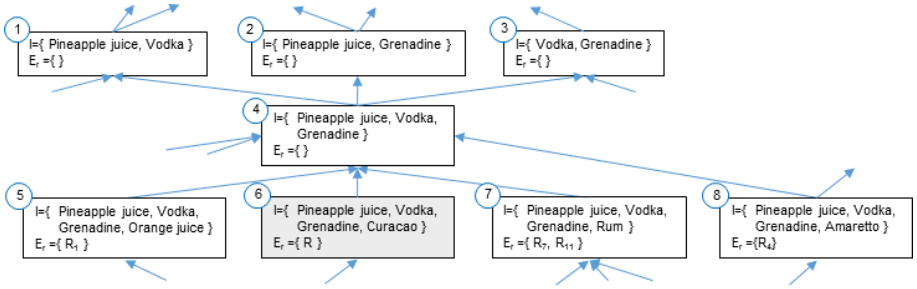

Fig. 3. Part of the concept lattice built from recipes using `PineappleJuice`, `Vodka`, and `Grenadine` (the ingredients that will be used in the resulting cocktail).

for *Cachaça* in the *Caipirinha* cocktail (in the intent of concept #11), some similar concepts (i.e. sharing a same super-concept) can be used. In the lattice given in example, concept #11 can be generalized to concept number #7, which extent contains cocktails with some alcohol, lime and some sugar. The cocktails in the extent of concept #12 are similar to the one of concept #11, because they share the `_Alcohol`, `Lime`, and `_Sugar` properties. When removing *"Cachaça"* from the *Caipirinha*, a possible ingredient for substitution, given by the lattice, could be `White rum`.

The approach exploiting the link between the concepts is used in many works using FCA for information retrieval. In Carpineto and Romano [6], the documents which are good answers to a query are searched in the lattice built from the document properties and from the query, around the concept representing the query. The same authors use this neighbour relation between concepts in a lattice for ordering documents returned by an information retrieval system [7].

Let C_R be the formal concept such that $E_r(C_R) = \{R\}$. A formal concept C close to C_R is searched according the following procedure. C is such that its intent $I(C)$ does not contain the substituting ingredient (`Curacao` in the example) and maximizes $|E_r(C)|$. First, C is searched in the ascendants of C_R, then in its siblings, and finally in the descendants of the siblings. The ingredient to be substituted is replaced by $I(C) \setminus I(C_R)$.

3.3 Real example of food substitution using FCA

To implement our approach, data about ingredient combinations in cocktail recipes has been collected. For this, we queried *Yummly* (`http://www.yummly.com/`). 16 queries were submitted; each query was composed of one ingredient (one available food) and was parametered to return all the *Yummly* cocktails and beverage recipes containing this ingredient. 9791 recipes have been collected. Unfortunately, the *Yummly* search engine does not necessarily return answers satisfying the query. So, the results are filtered, only to keep recipes that use at least one available food. Afterwards, the remaining recipes are deduplicated. After filtering and deduplicating, 6114 recipes are available, but only 1327 of them combine at least 2 available foods.

We show now, with query (3), how, after proposing to replace `OrangeJuice` with `PineappleJuice` and `StrawberrySyrup` with `Grenadine` in `R`, TAAABLE searches to replace `Curacao` which is not in the set of available foods. A part of the lattice resulting from the binary table containing recipes with `PineappleJuice`, `Grenadine` and `Vodka` is given in Fig. 3. Concept #6 corresponds to `R`, the recipe that must be adapted, and which has been added in the binary table to appear in the lattice. The most similar ingredient combination which includes `PineappleJuice`, `Grenadine` and `Vodka` is given by concept #7. Indeed, concept #8 cannot be used to produce a substitution because its intent contains `Amaretto` which is not an available food. Concept #5 intent contains `OrangeJuice`, an available food, but concept #5 is less close to concept #6 than concept #7, according to the selection procedure based on the maximal number of objects of E_r.

3.4 Adaptation of quantities with mixed integer linear optimization

Let us consider the following adaptation problem:

$\texttt{Source} =$

Recipe "Eggnog" (10 glasses)
10 cℓ of armagnac, 25 cℓ of rum, half a liter of milk, 5 eggs, 125 g of granulated sugar, 25 cℓ of fresh cream

$\texttt{Q}$ = "I want a cocktail recipe with cream but without egg or armagnac."

for which TUUURBINE produces the following ingredient substitution:

substitute egg and armagnac **with** banana and kirsch (4)

It must be noticed that this example does not comply with the constraints of the cocktail challenge (banana is not an available food), but has been chosen in order to illustrate various ideas related to adaptation of quantities. The approach to ingredient quantity adaptation is based on belief revision [8], applied to a formalization suited to adaptation of quantities. First, the adaptation problem $(\texttt{Source}, \texttt{Q})$ and the domain knowledge `DK` are formalized. Then, this adaptation process is described.

Formalization. Numerical variables are introduced to represent the ingredient quantities in a recipe. For the example, the following variables are introduced, for each food class `C`: $\texttt{alcohol}_\texttt{C}$, $\texttt{mass}_\texttt{C}$, $\texttt{number}_\texttt{C}$, $\texttt{sugar}_\texttt{C}$ and $\texttt{volume}_\texttt{C}$, which represent, respectively, the quantity (in grams) of alcohol in the ingredient `C` of the recipe, its mass (in grams), its number, its quantity (in grams) of sugar and its volume (in centiliters).[1] Therefore, the retrieved recipe can be expressed in this formalism by:

$$\begin{aligned}\texttt{Source} = {} & (\texttt{volume}_{\texttt{Armagnac}} = 10) \wedge (\texttt{volume}_{\texttt{Rum}} = 25) \wedge (\texttt{volume}_{\texttt{Milk}} = 50) \\ & \wedge (\texttt{number}_{\texttt{Egg}} = 5) \wedge (\texttt{mass}_{\texttt{GranulatedSugar}} = 125) \\ & \wedge (\texttt{volume}_{\texttt{FreshCream}} = 25)\end{aligned} \quad (5)$$

[1] One could consider other variables, e.g., the calories of ingredients, which would make possible to add constraints on the total number of calories in a dish.

In theory, all the variables could be continuous (represented by floating-point numbers). However, this can lead to adapted cases with, e.g., $\texttt{number}_{\texttt{Egg}} = 1.7$, which is avoided in most recipe books! For this reason, some variables v are declared as integer (denoted by $\tau(v) = \texttt{integer}$), the other ones as real numbers (denoted by $\tau(v) = \texttt{real}$).

The domain knowledge DK consists of a conjunction of *conversion equations*, *conservation equations* and *sign constraints*. The following conversion equations state that one egg without its shell has (on the average) a mass of 50 g, a volume of 5.2 cℓ, a quantity of sugar of 0.77 g and no alcohol:

$$\begin{array}{rr} \texttt{mass}_{\texttt{Egg}} = 50 \times \texttt{number}_{\texttt{Egg}} & \texttt{volume}_{\texttt{Egg}} = 5.2 \times \texttt{number}_{\texttt{Egg}} \\ \texttt{sugar}_{\texttt{Egg}} = 0.77 \times \texttt{number}_{\texttt{Egg}} & \texttt{alcohol}_{\texttt{Egg}} = 0. \end{array} \tag{6}$$

with $\tau(\texttt{mass}_{\texttt{Egg}}) = \tau(\texttt{volume}_{\texttt{Egg}}) = \tau(\texttt{sugar}_{\texttt{Egg}}) = \tau(\texttt{alcohol}_{\texttt{Egg}}) = \texttt{real}$ and $\tau(\texttt{number}_{\texttt{Egg}}) = \texttt{integer}$.

The following equations are also conjuncts of DK and represent the conservation of masses, volumes, etc.:

$$\begin{aligned} \texttt{mass}_{\texttt{EggOrEquivalent}} &= \texttt{mass}_{\texttt{Egg}} + \texttt{mass}_{\texttt{Banana}} \\ \texttt{volume}_{\texttt{Food}} &= \texttt{volume}_{\texttt{Liquid}} + \texttt{volume}_{\texttt{SolidFood}} \\ \texttt{volume}_{\texttt{Liquid}} &= \texttt{volume}_{\texttt{Brandy}} + \texttt{volume}_{\texttt{Rum}} + \texttt{volume}_{\texttt{FreshCream}} + \ldots \\ \texttt{volume}_{\texttt{Brandy}} &= \texttt{volume}_{\texttt{Armagnac}} + \texttt{volume}_{\texttt{Kirsch}} + \ldots \end{aligned} \tag{7}$$

where Food is the class of the food (any ingredient of a recipe is an instance of Food) and, e.g., $\texttt{alcohol}_{\texttt{Rum}}$ is related to $\texttt{volume}_{\texttt{Rum}}$ thanks to the conversion equation $\texttt{alcohol}_{\texttt{Rum}} = 0.4 \times \texttt{volume}_{\texttt{Rum}}$. Actually, equation (7) corresponds to the substitution of eggs by bananas.

Such conservation equations can be acquired using parts of the food hierarchy, thanks to some additional information. For instance, if C is a class of the hierarchy and $\{D_1, D_2, \ldots, D_p\}$ is a set of subclasses of C forming a partition of C (i.e., for each individual x of C, there is exactly one $i \in \{1, 2, \ldots, p\}$ such that x belongs to D_i), then $\texttt{mass}_C$ (resp., $\texttt{volume}_C$, $\texttt{number}_C$, etc.) is equal to the sum of the $\texttt{mass}_{D_i}$'s (resp., of the $\texttt{volume}_{D_i}$'s, of the $\texttt{number}_{D_i}$'s, etc.).

Finally, each variable v is assumed to satisfy the sign constraint $v \geqslant 0$.

The substitution (4) indicates that there should be neither egg nor armagnac in the adapted recipe. By contrast, there should be some bananas and kirsch but this piece of information can be entailed by the conservation equations. Therefore, the query is simply modeled by:

$$\texttt{Q} = (\texttt{mass}_{\texttt{Egg}} = 0) \wedge (\texttt{mass}_{\texttt{Armagnac}} = 0) \tag{8}$$

The adaptation problem is now formalized: the source case is formalized by (5); the query is formalized by (8) and the domain knowledge is given by the conversion and conservation equations, and the sign constraints. Since the source case and the query are to be understood wrt the domain knowledge, the formulas for them are, respectively, $\texttt{DK} \wedge \texttt{Source}$ and $\texttt{DK} \wedge \texttt{Q}$. The result of the adaptation will be denoted by AdaptedCase.

Description of the adaptation process. Let $\{v_1, v_2, \ldots, v_n\}$ be the set of the variables used in `Source`, `Q` and `DK`. In the representation space based on the formalism used above, a particular recipe is represented by a tuple $x = (x_1, x_2, \ldots, x_n) \in \Omega$, where $\Omega = \Omega_1 \times \Omega_2 \times \ldots \times \Omega_n$ such that $\Omega_i = \mathbb{Z}$ if $\tau(v_i) =$ `integer` and $\Omega_i = \mathbb{R}$ otherwise ($\mathbb{R}$: set of real numbers, $\mathbb{Z}$: set of integers). Given φ, a conjunction of linear constraints, let $\mathcal{M}(\varphi)$ be the set of $x \in \Omega$ such that x verifies all the constraints of φ. The function $\varphi \mapsto \mathcal{M}(\varphi)$ provides a model-theoretical semantics to the logic of the conjunction of linear constraints: φ_1 entails φ_2 if $\mathcal{M}(\varphi_1) \subseteq \mathcal{M}(\varphi_2)$.

The principle of revision-based adaptation consists in a minimal modification of $\texttt{DK} \wedge \texttt{Source}$ so that it becomes consistent with $\texttt{DK} \wedge \texttt{Q}$. Such a minimal modification can be computed thanks to a belief revision operator based on a distance function d on Ω, meaning that the modification from an $x \in \Omega$ to an $y \in \Omega$ is measured by $d(x, y)$. Let $S = \mathcal{M}(\texttt{DK} \wedge \texttt{Source})$ and $Q = \mathcal{M}(\texttt{DK} \wedge \texttt{Q})$. The minimal modification from the source case to the query is therefore measured by $d^* = d(S, Q) = \inf_{x \in S, y \in Q} d(x, y)$. Thus, `AdaptedCase` is such that

$$\mathcal{M}(\texttt{AdaptedCase}) = \{y \in Q \mid d(S, y) = d^*\}$$

where $d(S, y) = \inf_{x \in S} d(x, y)$.

Now, d is assumed to be a Manhattan distance function:

$$d(x, y) = \sum_{i=1}^{n} w_i |y_i - x_i|$$

where $w_i > 0$ is a weight associated to the variable v_i. Such a weight captures the effort of change for this variable. For example, if $v_i = \texttt{volume}_{\texttt{LemonJuice}}$ and $v_j = \texttt{volume}_{\texttt{Vodka}}$, then $w_i < w_j$ means that the adaptation process is less "reluctant" to change the volume of lemon juice than to change the volume of vodka.

Under this assumption, $\mathcal{M}(\texttt{AdaptedCase})$ is the solution of the following optimization problem in y:

$$x \in \mathcal{M}(\texttt{DK} \wedge \texttt{Source}) \qquad y \in \mathcal{M}(\texttt{DK} \wedge \texttt{Q}) \tag{9}$$

$$\text{minimize } d(x, y) \tag{10}$$

The conjunctions of constraints (9) are linear but the objective function (10) is not. Now, it can be shown that the set of solutions to this problem coincides with the set of solutions to the following optimization problem in y:

$$x \in \mathcal{M}(\texttt{DK} \wedge \texttt{Source}) \qquad y \in \mathcal{M}(\texttt{DK} \wedge \texttt{Q})$$

$$\bigwedge_{i=1}^{n} z_i \geq y_i - x_i \qquad \bigwedge_{i=1}^{n} z_i \geq x_i - y_i$$

$$\text{minimize } \sum_{i=1}^{n} w_i z_i$$

which is linear, and thus can be solved with classical operational research techniques. It is noteworthy that if every variable is continuous, then this optimization problem is polynomial, otherwise, it is a mixed integer linear optimization, known to be an NP-hard problem. In practice, the more variables are integers, the more it will require computing time; thus, if a variable range is big enough, it may be more appropriate to consider it as real. The heuristic we have chosen is as follows. If, for a type of food F, it appears in all the recipes of the case base as units, then $\tau(\texttt{number}_F) = \texttt{integer}$.

When this linear problem is solved, this gives a solution to the query, expressed with all the n variables. From a human-interface viewpoint, some of these variables should not be displayed. For example, if an ingredient is given by its volume in the source recipe, then it should not be given as a mass in the adapted case. Since DK relates masses to volumes, there is no loss of information.

With the example presented above, the result is as follows:

$$\begin{aligned}\texttt{AdaptedCase} \equiv\ & \texttt{DK}\\ & \wedge (\texttt{volume}_{\texttt{Kirsch}} = 9) \wedge (\texttt{volume}_{\texttt{Rum}} = 25) \wedge (\texttt{volume}_{\texttt{Milk}} = 50)\\ & \wedge (\texttt{number}_{\texttt{Banana}} = 2) \wedge (\texttt{mass}_{\texttt{GranulatedSugar}} = 96)\\ & \wedge (\texttt{volume}_{\texttt{FreshCream}} = 290)\end{aligned}$$

It can be noticed that AdaptedCase entails $\texttt{DK} \wedge \texttt{Q}$, which was expected. For this example, the following weights have been chosen assuming that more a variable correponds to a general concept more its associated weight has to be large:

$$w_{\texttt{volume}_{\texttt{Food}}} = 100 \qquad w_{\texttt{sugar}_{\texttt{Food}}} = 50 \qquad w_{\texttt{alcohol}_{\texttt{Food}}} = 50$$
$$w_{\texttt{volume}_{\texttt{Brandy}}} = 5 \qquad w_{\texttt{mass}_{\texttt{EggOrEquivalent}}} = 10$$
$$\text{and } w_v = 1 \text{ for any other variable } v$$

Translated back in an informal way, this gives:

AdaptedCase =	Recipe “Eggnog” (10 glasses) after adaptation
	9 cℓ of kirsch, 25 cℓ of rum, half a liter of milk, 2 bananas, 96 g of sugar, 290 cℓ of fresh cream

This result illustrates the quantity compensations done by the adaptation: the quantity of sugar has been lowered because bananas are sweeter than eggs and the volume of kirsch is higher than the volume of armagnac in the source recipe, because the degree of alcohol is lower for armagnac than for kirsch.

4 Sandwich challenge

The *sandwich challenge* is addressed with the 2014 Taaable system [9], which is efficient for the ingredient susbtitution step. The preparation procedure of the adapted recipe uses, in the same order, the steps used in the source recipe, because the ontology-based substitution procedure of Taaable favors the substitution of ingredients of the same type (e.g., a sauce by a sauce). So, the order of the ingredients in the adapted recipe will be the same as in the source recipe.

To adapt the textual preparation of the recipe, the text occurrences of the replaced ingredients are substituted with the replacing ingredients. A set of rules allows to identify plurals of the removed ingredient in the text, and replace them with the plural form of the replacing ingredients. For example, when replacing *mayo* with *mustard*, "Apply *mayo* on one slice, tomato sauce on the other." is adapted to "Apply *mustard* on one slice, tomato sauce on the other."

5 Conclusion

This paper has presented the two systems developed by the Taaable team for its participation to the 2015 CCC. The two systems are based on the previous version of Taaable, extended with two new approaches: a FCA approach to guide ingredient substitution, and an adaptation of the ingredient quantities based on a mixed linear optimization. The work presented here still needs a thorough evaluation: ongoing work addresses this issue, following the methodology introduced in [2].

References

1. A. Cordier, V. Dufour-Lussier, J. Lieber, E. Nauer, F. Badra, J. Cojan, E. Gaillard, L. Infante-Blanco, P. Molli, A. Napoli, and H. Skaf-Molli. Taaable: a Case-Based System for personalized Cooking. In S. Montani and L. C. Jain, editors, *Successful Case-based Reasoning Applications-2*, volume 494 of *Studies in Computational Intelligence*, pages 121–162. Springer, 2014.
2. E. Gaillard, J. Lieber, E. Nauer, and A. Cordier. How Case-Based Reasoning on e-Community Knowledge Can Be Improved Thanks to Knowledge Reliability. In *Case-Based Reasoning Research and Development*, volume 8765, pages 155 – 169, Cork, Ireland, Ireland, September 2014. L. Lamontagne and E. Plaza.
3. E. Gaillard, L. Infante-Blanco, J. Lieber, and E. Nauer. Tuuurbine: A Generic CBR Engine over RDFS. In *Case-Based Reasoning Research and Development*, volume 8765, pages 140 – 154, Cork, Ireland, September 2014.
4. E. Gaillard, J. Lieber, and E. Nauer. Adaptation knowledge discovery for cooking using closed itemset extraction. In *The Eighth International Conference on Concept Lattices and their Applications - CLA 2011*, pages 87–99, 2011.
5. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin, 1999.
6. C. Carpineto and G. Romano. Effective Reformulation of Boolean Queries with Concept Lattices. In T. Andreasen, H. Christiansen, and H. Legind Larsen, editors, *Flexible Query Answering Systems, Third International Conference (FQAS'98)*, volume 1495 of *LNCS*, pages 83–94. Springer, 1998.
7. C. Carpineto and G. Romano. Order-Theoretical Ranking. *Journal of the American Society for Information Science*, 51(7):587–601, 2000.
8. J. Cojan and J. Lieber. Applying Belief Revision to Case-Based Reasoning. In H. Prade and G. Richard, editors, *Computational Approaches to Analogical Reasoning: Current Trends*, volume 548 of *Studies in Computational Intelligence*, pages 133 – 161. Springer, 2014.
9. E. Gaillard, J. Lieber, and E. Nauer. Case-Based Cooking with Generic Computer Utensils: Taaable Next Generation. In *Proceedings of the ICCBR 2014 Workshops*, number pp 89-100, page 254, Cork, Ireland, 2014. D. B. Leake and J. Lieber.

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

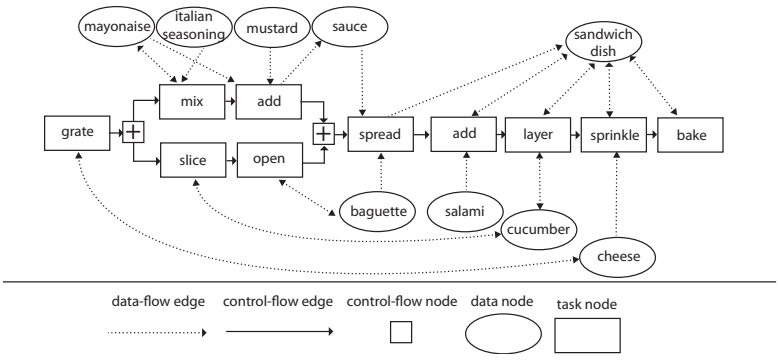

Fig. 1. Example of a block-oriented cooking workflow

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.

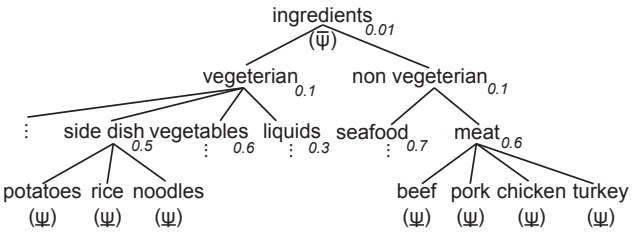

Fig. 2. Example of an ingredient taxonomy

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.

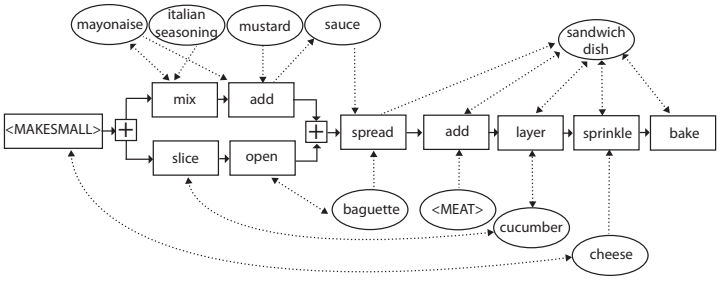

Fig. 3. Example of a generalized workflow

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.

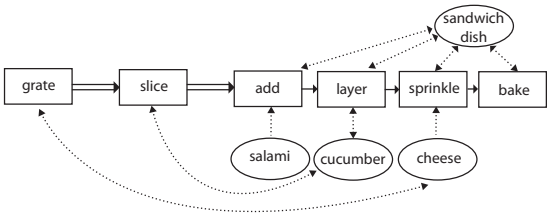

Fig. 4. Example of a workflow stream

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.

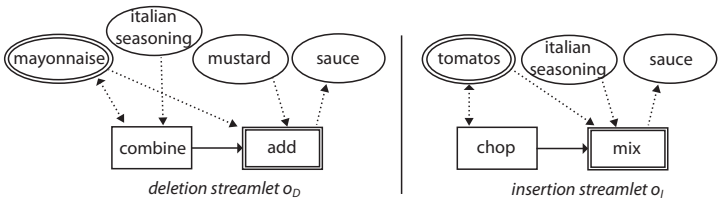

Fig. 5. Example of a workflow adaptation operator

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.

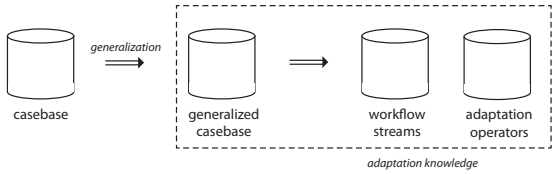

Fig. 6. Generation of adaptation knowledge

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.

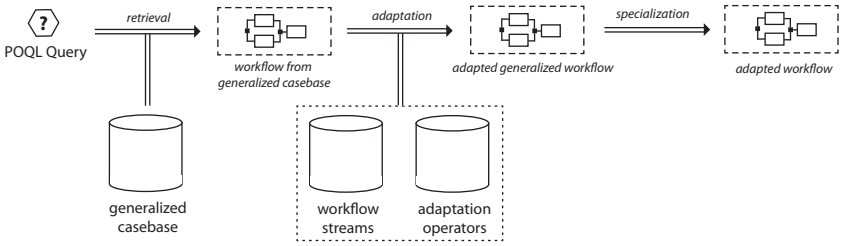

Fig. 7. Workflow adaptation

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-

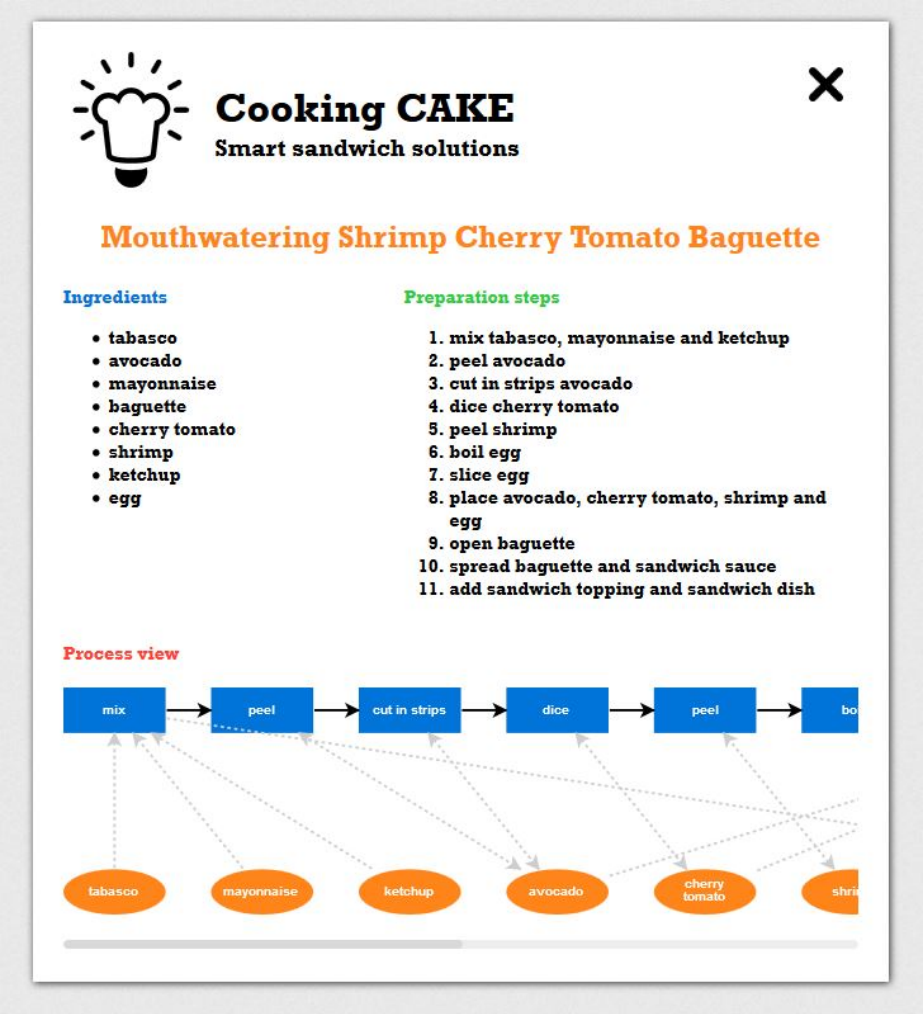

Fig. 9. Example recipe generated by CookingCAKE

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)

≪CooCo, what can I cook today? Surprise me.≫

Karen Insa Wolf, Stefan Goetze, and Frank Wallhoff

Fraunhofer Institute for Digital Media Technology IDMT
Marie-Curie-Strasse 2, 26129 Oldenburg, Germany
{insa.wolf,stefan.goetze,frank.wallhoff}@idmt.fraunhofer.de
http://www.idmt.fraunhofer.de

Abstract. In this paper a heuristic computer-based approach is described to vary cooking recipes by replacing ingredients. Conceptually, the approach is integrated in a speech dialogue system. The approach is based on a scoring system. The score value is used to rate different ingredients as candidates to substitute a specific ingredient of a recipe. This substitution score depends on different factors: 1) rating of the similarity between the ingredient which has to be replaced and the substitution candidate 2) rating how well the substitution candidate fits the recipe 3) gustatory preferences of the user. The substitution candidate with the highest score is proposed to the user.

Keywords: speech dialogue system, cooking coach, recipe variation

1 Motivation

The task in the open challenge of the Computer Cooking Contest [1] is a computer-based adaptation of cooking recipes. The present contribution proposes an approach to substitute ingredients of recipes. The approach is integrated in a speech dialogue system, called CooCo (Cooking Coach), introduced in [13]. CooCo is currently being further developed. A speech dialogue system is a suitable framework for this task:

- Speech input and output is a natural and convenient way to interact with technical devices or systems.
- A speech dialogue system is particularly suitable in scenarios in which the user cannot use his or her hands for interaction. Keyboard, mouse or touchscreen are not convenient user interfaces while cooking.
- Assuming a flexible dialogue management, spontaneous utterance of the user (like e.g. ≪Oops, I do not have ...≫) can be processed.
- The user can be involved in a unobtrusive manner to improve the recipe variation result and tailor the recipe to her/his personal gusto.

2 Concept of CooCo

CooCo is designed to assist users in different scenarios: The user can ask for recipes while doing the dishes or can get reminders regarding timing and next

steps while cooking. Both tasks require a context-based dialogue system including modules for interpreting, planning and re-planning, as well as memorizing and learning. Different approaches to realize a speech dialogue manager exist, e.g. [9]. Lison distinguishes between hand-crafted and statistical approaches and proposes the toolkit OpenDial to combine both [8]. The dialogue manager of CooCo is based on OpenDial [10]. CooCo's assistance while cooking is conceptually based on a dynamic planning module to actively manage the cooking process. This goes beyond simply reading out the cooking steps aloud when the user asks for this [11]. CooCo formulates an action plan considering active and passive time of the user (e.g. cutting vs. simmering) and dependencies of the cooking steps [13]. The recipe advice mode includes generic models of gustatory preferences (e.g. hot or sweet depending on typical amount of ingredients like chili or sugar) which will be adapted based on the feedback of the user. A new feature, presented in this paper, is the variation of the cooking recipes.

3 Computer-based variation of cooking recipes

The computer-based variations of cooking recipes addresses topics of artificial intelligence and machine learning approaches. The task to derive the consequences of the substitution of an ingredient on the textual description of the preparation steps requires techniques of natural language understanding, e.g. [2]. Other approaches aim at replacing ingredients, e.g. by randomizing recipe items [3], by using cognitive super computing (based on IBM's computer system WATSON, [6]) or by just enlarging the database (by the help of a community) to find a matching recipe for every combination of ingredients [12].

The approach presented here addresses the replacement of ingredients. Thereby, I_{db} is the set of all ingredients (I) of a specific database. A subset $I_{rc} \subseteq I_{db}$ with ingredients, which belong to one recipe, is defined as $I_{rc} = \{i_{rc,1}, \ldots, i_{rc,m}\}$ with maximum number m of ingredients. The subset $I_{sb} = \{i_{sb,1}, \ldots, i_{sb,h}\}$ with $h \leq m$ and $I_{sb} \subseteq I_{rc}$ comprises all ingredients which will be substituted. The food items which are candidates (C) to substitute one element of I_{sb} belong to the set $C_{sb} = \{c_{sb,1}, ..., c_{sb,n}\}$ with maximum number n of known food items. The set of the remaining ingredients of the recipe without the elements of I_{sb} is defined as $I_{rm} = I_{rc} \setminus I_{sb}$. The approach is based on the computation of a substitution score s ranging from 0 to 120 indicating the fit of a specific substitution pair $i_{sb,j} \in I_{sb}$ and $c_{sb,k} \in C_{sb}$. The substitution score is based on statistical information derived from a recipe database and general food knowledge. The approach can also be regarded as one module of a case-based reasoning process of a recipe advisor, as it is described e.g. in [7], to include the substitution of ingredients.

4 Use cases

The central task in the following two use cases is to propose a tasty recipe based on the user's input by replacing ingredients. The intention of the user differs in

the scenarios. Both use cases can be extended by including the question of undesired ingredients. In order to enlarge the number of possible recipe candidates, the proposed recipe variation approach can be applied in this case additionally to substitute undesired ingredients. Users differ in their gustatory preferences, one likes more traditional recipes, while the other is more open to new tastes. To adjust these individual preferences, two user parameters are introduced referred to as experimental levels. The experimental level e_{cd} influences how common or uncommon a substitution candidate should be. The level e_{cb} regulates how common or uncommon the combination of a substitution candidate and all elements of I_{rm} is. For both levels three adjustment steps can be chosen by the user, ranging from 1 = very common to 3 = very uncommon.

4.1 Use case 1: ≪Suprise me.≫

Based on one chosen recipe the user asks for a variation of this recipe. A similar scenario would be that the user realizes that one ingredient is missing but s/he still wants to cook the chosen recipe accepting variations. In both cases, CooCo can choose freely possible substitution candidates. In the first case, the ingredient $i_{sb,j}$ is not defined by the user. In the second case, $i_{sb,j}$ is the missing ingredient.

4.2 Use case 2: ≪Work with what I have.≫

The user specifies some ingredients I_{us}, s/he wants to work with, but no recipe can be found in the database which uses all desired ingredients. The task for CooCo is now to propose one recipe which matches by replacing missing elements (I_{ms}) of I_{rc} with those of I_{us}. For this scenario, a plausibility check is necessary since not each combination of ingredients presents a suitable option for a recipe.

5 CooCo's Recipe Variation Approach

The central aim of the approach is to compute substitution scores s for different substitution candidates of C_{sb} in relation to one element $i_{sb,j}$ of I_{sb}. The candidate $c_{sb,k}$ with the highest score is finally proposed to the user. Considering the abbreviations $i_{sb,j} = i$ and $c_{sb,k} = c$ the substitution score $s(i, c)$ is derived as

$$s(i,c) = s_b(i,c) + s_{sp}(i,c) + s_n(i,c) + s_{cd}(c) + s_{cb}(c, i_{rm}|i_{rm} \in I_{rm}), \qquad (1)$$

with s_b as basic substitution score, s_{sp} as special substitution score, s_n as substitution score based on nutrition facts, and s_{cd} and s_{cb} as substitution scores derived from a statistical analysis of the ingredients and their combination frequency based on the recipe database. The derivation of each summand of Eq. 1 is explained in the following. The substitution of more than one ingredient can be done by repeating the algorithm, up to now without considering the results of subsequent substitution steps. As starting point a recipe database with 1.222 recipes is chosen [5]. Additionally, a semantic net is created representing food

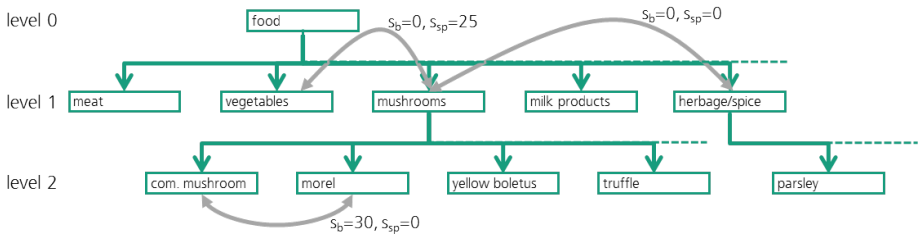

Fig. 1. Part of the semantic net with substitution scores s_b and s_{sp}.

items in a structured way, cf. Fig. 1. Each item is represented as class within a relationship network of currently 120 classes starting from the level 0 up to 3.

Besides the parent-children constellation different properties of each food class are stored. These properties are grouped in (a) those properties considering only the class itself and (b) those properties related to other classes. For group (a), the following properties are introduced:

nutrition facts n_g, with $g = \{c, f, p, e\}$: Nutrition facts are stored for different food classes of level 2 or higher. In the first version of CooCo, the variable n_c contains carbohydrates, n_f fat,n_p protein, and n_e energy per 100 g.

relative frequency f_{cd}: For each food class its relative frequency is derived based on the recipe database. The number of recipes in which the class occurs as ingredient is divided by the total number of recipes. This frequency value describes how common or uncommon a certain ingredient is.

substitution score s_{cd}: Based on the relative frequency f_{cd} the score s_{cd} is derived, considering the experimental level e_{cd}. The relative frequencies f_{cd} are classified in five categories. The first category $D_{cd,1}$ contains rarely used and the last category $D_{cd,5}$ frequently used ingredients, assuming $D_{cd} := [0 \ldots 0.005 \ldots 0.01 \ldots 0.03 \ldots 0.08 \ldots 1.0]$. The index of the category in combination with the experimental level e_{cd} defines the magnitude of the substitution score s_{cd}. This is implemented using a weighting matrix

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \end{bmatrix} = \begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & -1 & -2 \end{bmatrix} \tag{2}$$

and by taking one of its elements to derive

$$s_{cd} = 10 w_{e_{cd},p}, \tag{3}$$

with row number e_{cd} and column number p as index of $D_{cd,p}$.

The group (b) of properties considers the relation between two food classes to indicate how good they can substitute each other or how good they can be combined in one recipe.

basic substitution score s_b: It is assumed that food classes at a low semantic level (e.g. common mushrooms, morel or truffel in the class "mushrooms", level 2) are similar to each other. Therefore, the score s_b is introduced depending on the level of class in the semantic net, cf. Fig. 1.

special substitution score s_{sp}: A few explicitly defined scores s_{sp} are stored (e.g. vegetables as substitution candidate for mushrooms or the milk product "tofu" as good substitution candidate for children of the class "meat").

substitution score based on nutrition facts s_n: It is assumed that one food class of level 2 or higher is a good substitution candidate for another food class if they have similar nutrition facts. Therefore, the similarity factor f_n of two ingredients i_A and i_B is derived based on their nutrition facts n_g as

$$f_{n,g}(i_A, i_B) = \frac{|n_g(i_A) - n_g(i_B)|}{(n_g(i_A) + n_g(i_B))}, \tag{4}$$

with $g = \{c, f, p, e\}$. The mean value μ_{f_n} and the standard deviation σ_{f_n} derived from all $f_{n,g}$ is used as measure of the similarity of the nutrition facts - being aware of the roughness and simplicity of this approach. The substitution score based on nutrition facts is derived as

$$s_n = \min(2/\mu_{f_n}, 15) + \min(2/\sigma_{f_n}, 15). \tag{5}$$

relative combination frequency f_{cb}: The frequency value $f_{cb}(i_A, i_B)$ expresses how often an ingredient i_A is used in combination with a specific ingredient i_B of I_{db}. Therefore, the number of recipes $n_{A\&B}$, in which both ingredients i_A and i_B are included, is determined. This yields $f_{cb}(i_A, i_B) = n_{A\&B}/n_A$. As the denominator usually differs numerically for $f_{cb}(i_A, i_B)$ and $f_{cb}(i_B, i_A)$, the frequencies differ correspondingly. Following this approach, it has to be considered that uncommon ingredients can get f_{cb} values close to 1.0 as n_A is close to $n_{A\&B}$.

substitution score s_{cb}: The score s_{cb} is derived in a similar way as it is done for s_{cd}, but now considering the relative frequency f_{cb} and the experimental level e_{cb}. All frequencies $f_{cb}(c, i)$ of the substitution candidate $c \in C_{sb}$ and the ingredients $i \in I_{rem}$ have to be calculated first. The mean value of all these frequencies is then derived as $f_{cb}(c, I_{rm})$. This value is finally assigned to one of the categories $B_{cb,1}$ up to $B_{cb,5}$, heuristically defined as $B_{cb} := [0 \ldots 0.30 \ldots 0.35 \ldots 0.40 \ldots 0.50 \ldots 1.0]$. Using the weighting matrix W from Eq. 2 the score is derived as

$$s_{cb} = 10 w_{e_{cb},q}, \tag{6}$$

with row number e_{cb} and column number q as index of $B_{cb,q}$. The effect is, that if the user wants uncommon combinations, expressed as $e_{cb} = 3$, a set of ingredients with a low $f_{cb}(c, I_{rm})$ get a high score.

6 Implementation

The knowledge base containing the recipe database and the semantic net are implemented in Prolog. Standard request functions are implemented, so that recipes

including or excluding specific ingredients can be looked up. The approach described above is conceptually tested, further implementation and evaluation is on going work. The procedures to handle both use cases (cf. Section 4) are described in the following based on one test example. The starting point is a simple mushroom soup recipe:

250 g common mushrooms,	40 g butter,	40 g flour,
5 dl bouillon,	5 dl milk,	1 tb parsely, minced,
- - salt,	- - pepper	

6.1 Procedure for use case 1: ≪Surprise me.≫

In the following description only some examples of the possible substitution candidates are listed.

1. Choose the ingredient with the largest proportion relative to all ingredients of I_{rc} as i_{sb} [*common mushrooms*].
2. Compute $(s_b + s_{sp})$ for all $c_{sb} \in C_{sb}$ [class "mushrooms": *yellow boletus, morel, truffle*; extract of class "vegetables": *red pepper, tomato, cucumber*]
3. Compute s_n for all pairs of c_{sb} and i_{sb} [listing *parsley, cauliflower, morel, yellow boletus* as the one with a high score s_n].
4. Derive s_{cd} for all c_{sb}.
5. Derive s_{cb} for all c_{sb} with respect to the elements of I_{rem}.
6. Sum up s for each pair of c_{sb} and i_{sb} following Eq. 1.

Numerical results for the different experimental levels e_{cd} and e_{cb} are listed in Tab. 1. Parsley is left out as it is already part of the recipe. The results show, that a user with a low e_{cd} of 1 and a medium or high e_{cb} of 2 or 3 will be recommended a tomato soup. In case a very common combination of ingredients is wanted $(s(1,1))$, morel soup is proposed instead. Reason for this is that the recipes with common mushrooms and morel often share the basic combination of ingredients. A user who wants uncommon ingredients in a uncommon combination gets truffle as substitution candidate $(s(3,3))$. Elements of the class mushrooms are mostly preferred. A whole class like "mushrooms" could also be excluded, resulting in recommendations of cauliflower as substitution candidate as a less common ingredient than tomatoes. As the terms s_{cd} and s_{cb} are based on the statistical analysis of the recipe database, the result depends strongly on the size and the quality of the recipe database.

6.2 Procedure for use case 2: ≪Work with what I have.≫

In use case 2 the user desires a recipe with the ingredient set $I_{us} = \{$*butter, flour, parsley, bouillon, red pepper*$\}$. Firstly, for all elements of I_{us} the frequencies f_{cb} to each other are checked heuristically: If all $f_{cb} > 0$ and the mean $\mu_{f_{cb}} > 0.1$, then CooCo accepts the set I_{us}. Otherwise the dialogue with the user is reopened to ask for other set members. In the example, the set is accepted. The recipe that matches best I_{us} is mushroom soup, based on the simple rule to look for

Table 1. Numerical results of use case 1. The result $s(j,k)$ means s based on $e_{cd} = j$ $e_{cb} = k$. The respective candidate with the largest score s is marked in bold letters.

	y. boletus	morel	truffle	red pepper	tomato	cucumber	cauliflower
$s_b + s_{sp}$	30	30	30	25	25	25	25
s_n	29.2	29.2	29.0	9.4	5.6	12.8	33.0
f_{cd}	0.007	0.004	0.002	0.107	0.142	0.010	0.010
f_{cb}	0.397	0.543	0.286	0.347	0.346	0.236	0.367
$s(1,1)$	49	**59**	19	44	51	8	48
$s(1,2)$	49	39	39	54	**61**	28	48
$s(1,3)$	49	19	59	64	**71**	48	48
$s(2,1)$	59	**79**	39	24	31	18	58
$s(2,2)$	59	**59**	**59**	34	41	38	58
$s(2,3)$	59	39	**79**	44	51	58	58
$s(3,1)$	69	**99**	59	4	11	28	68
$s(3,2)$	69	**79**	**79**	14	21	48	68
$s(3,3)$	69	59	**99**	24	31	68	68

those recipes with the smallest number of missing ingredients $I_{ms} = \{i_{ms} | (i_{ms} \in I_{rc}) \wedge (i_{ms} \notin I_{us})\}$. However, red pepper is not part of the original recipe. The algorithm now attends to compute based on Eq. 1 as criteria whether red pepper is a suitable substitution candidate c_{sb} for one of the missing ingredients. Some of the missing ingredients {*pepper, salt, bouillon*} are marked as standard ingredients in the database. CooCo assumes as first guess that they are available also in case the user did not mentioned them explicitly. If this is confirmed by the user, the only missing ingredients left are $I_{ms} = \{$*common mushrooms, milk*$\}$. Considering the experimental levels, the score s is derived for all pairs of c_{sb} with one of the elements of I_{ms}. The computation result looking at the substitution pair *red pepper - common mushrooms* differs slightly compared to the result of use case 1 because *milk* is left out in the computation of s_{cb} as it is missing. As consequence, f_{cb} is classified here in $B_{cb,3}$ resulting in $s_{cb} = 0$, independently of e_{cb} as the weight factor in Eq. 6 is zero. Therefore, for all e_{cb} levels the score s is identical to $s(j,2)$ with $e_{cd} = j$, cf. Tab. 1. The highest score $s = 54$ is reached for $e_{cd} = 1$. Considering a threshold scheme of $[120 \ldots 80]$ (very good), $]80 \ldots 40]$ (acceptable), $]40 \ldots 0]$ (not recommended) for s, the substitution pair *red pepper - common mushrooms* is evaluated as "acceptable". In no case it is an option to replace *milk* with *red pepper*, the highest score is $s = 29$. This is reasonable, but a rule should be added in future versions to avoid the substitution of liquid and solid ingredients in any case. This is possible by adding an appropriate property in the semantic net. Milk remains here as missing candidate. Two different last options are possible: (1) Ask the user explicitly whether there is after all a potential substitution candidate. If yes, repeat the procedure. (2) Evaluate how well the missing ingredient could be omitted. Therefore, $s_b + s_{sp} + s_n$ is computed in relation to all ingredients of I_{rm} to get a hint if one of them could make up for the omission by increasing its quantity. In this specific example, the result of 17.5 for *milk* in relation to *butter* is not promising enough to propose this

as solution. As final step, the amount of liquid within the recipe ingredients is checked leading here to an increase of the amount of bouillon to recover the original amount of liquid. The final solution with appropriate comments based on the score s is presented to the user.

7 Conclusion and future work

A new feature of the currently developed application CooCo is presented. An approach to derive recipe variations by replacing ingredients is introduced. Two different use cases are addressed. The introduced examples provide reasonable results. This first proposed version of the approach has to be further improved and expanded in future work. An evaluation of the substitution results is planned based on feedback of users integrated in the speech dialogue system. The mechanism how to choose the best starting recipe in use case 2 can be ameliorated, including e.g. more information of the gustatory preferences of the user. The present approach prefers recipes with a small number of ingredients. In summary, the approach is a first step for computer-based tasty cooking recipe variations.

Acknowledgement. The authors would like to thank the anonymous reviewers for their helpful comments to improve the final version of the paper and their suggestions for future work in this research field.

References

1. CCC: Computer cooking contest (2015), `cc2015.loria.fr/index.php`, 3.3.2015
2. Dufour-Lussier, V., Le Ber, F., Lieber, J., Nauer, E.: Automatic case acquisition from texts for process-oriented case-based reasoning. Inf Systems 40, 153–167 (2014)
3. Easier Baking, `www.easierbaking.com`, 9.7.2015
4. Gamrad, D.: Modeling, simulation, and realization of cognitive technical systems. Ph.D. thesis, Universität Duisburg-Essen (2011)
5. Herz, J.: Kalorio (2015), `www.kalorio.de/`, 27.2.2015
6. IBM, Inst of Culinary Education: Cognitive Cooking with Chef Watson: Recipes for Innovation from IBM & the Institute of Culinary Education. Sourcebooks (2015)
7. Ihle, N., Newo, R., Hanft, A., Bach, K., Reichle, M.: CookIIS - A Case-Based Recipe Advisor. In: Proc 8th Int Conf on CBR 2009, Seattle, USA, pp. 269-278 (2009)
8. Lison, P.: Structured Probabilistic Modelling for Dialogue Management. Ph.D. thesis, Dep Informatics, University of Oslo (2014)
9. Morbini, F., Audhkhasi, K., Sagae, K., Artstein, R., Can, D., Georgiou, P., Narayanan, S., Leuski, A., Traum, D.: Which ASR should I choose for my dialogue system? In: Proc SIGDIAL 2013, Metz, France. pp. 394 – 403 (2013)
10. Open Dial, `www.opendial-toolkit.net`, 9.7.2015
11. Schäfer, U., Arnold, F., Ostermann, S., Reifers, S.: Ingredients and recipe for a robust mobile speech-enabled cooking assistant for german. In: KI 2013: Advances in Artificial Intelligence, pp. 212–223. No. 8077 in LNCS, Springer (2013)
12. SousChef, `www.acaciatreesoftware.com/`, 9.7.2015
13. Wolf, K.I., Goetze, S., Wellmann, J., Winneke, A., Wallhoff, F.: Concept of a nutrition consultant application with context based speech recognition. In: Proc Workshop Kognitive Systeme 2015, Bielefeld (2015)

Enriching Cooking Workflows with Multimedia Data from a High Security Cloud Storage

Patrick Bedué | bedue@stud.uni-frankfurt.de
Wenxia Han | s0611400@stud.uni-frankfurt.de
Mathias Hauschild | s8722400@stud.uni-frankfurt.de
Maximilian Pötz | s8084343@stud.uni-frankfurt.de
Mirjam Minor | minor@cs.uni-frankfurt.de

Abstract: With increasing growth of cloud services and the ability to choose from different cloud providers, we propose a new way to connect cooking workflows with a high security cloud storage. We use Activiti for workflow design and JavaScript Object Notation (JSON) for structured data interchange with a sealed cloud storage. This approach supports cooking workflows with instructions from multimedia data (e.g. videos, pictures) for special interest groups of the cooking domain like private communities or even chronically ill patients. The paper makes a contribution to current trends in information-systems related research such as scalability and experience reuse. Further, it connects Cloud Computing with Business Process Management.

1 Introduction

With growing globalization, information technology has become a key resource for business success or failure. IT is often used to manage business processes in companies and has become increasingly important, leading to a rise in new ways to organize business processes (Aalst, Benatallah et al. 2003). Cloud Computing changes the way we can develop and organize our resources and also enables a flexible and individual allocation of resources (Ciovică, Cristescu et al. 2014, Schulte, Janiesch et al. 2015).

Business processes are modelled as a collection of activities, dependencies between activities and are technically supported by workflows (Aalst, Benatallah et al. 2003). Regarding actual research topics, realizing business processes in a flexible and cost-efficient way is on a rise (Schulte, Janiesch et al. 2015). There are some studies focusing on workflow execution in the cloud investigating infrastructural challenges of elastic Business Process Management or security issues (Wang, Korambath et al. 2014, Schulte, Janiesch et al. 2015).

To develop our research proposition, we concentrate our work on the implementation of workflows in the cooking domain with a special focus on applying secure cloud technology for multimedia data integration and data objects representing ingredients.

To improve the user experience, we investigate the feasibility of using multimedia data from a high security cloud storage and of integrating this data into workflows. The user advantage is that she no longer needs to store big multimedia files on her device and can execute workflows in a web interface. Secure cloud technology provides rapid

scalability as well as data protection. Both are beneficial properties for using multimedia data in the cooking domain. With our proposed solution, the user is able to easily create cooking recipes on a platform and store multimedia data. Having started the workflow, the user is provided with cooking instructions including pictures or videos. Using secure cloud technology addresses private communities which e.g. do not want to publish content or appear in any videos accessible to the public, for instance chronically ill persons or allergy sufferers who do not want to share their special recipes or videos because their identity might be linked to serious diseases. For many people it is not an option to simply store the content on scalable platforms like YouTube because their children might be visible or because their employer might get knowledge about their special concerns. Our solution combines scalability with additional security and ensures that content is protected from persons who are not supposed to have access to the recipes. The technical solution is based on IDGARD which allows to separate data in different, sealed containers and to keep own videos or pictures confidential.

With this new solution we address the open challenge of the Computer Cooking Contest.

The remainder of our paper is structured as follows. At first we will provide an overview of our architecture and a sample of a workflow to ease comprehensibility. After that we will present our implementation concept and workflows from the cooking domain, which are based on cooking recipes of pasta. We conclude with a discussion of our research contribution and the new prospects for both companies and researchers.

2 Fundamentals

Sealed Cloud

A broad range of providers offer different models for services at different layers. But most of them do not guarantee security for the clients of a cloud provider or the privacy of the data (Santos, Gummadi et al. 2009). Especially in Germany, data protection and compliance issues require new technologies such as sealed cloud technology (Rieken 2015). A sealed cloud offers the technology to encrypt contents and meta data so that the cloud provider itself is not able to access data contents. The monitoring of user behavior and the possible access to protected content by providers is a big issue as part of data privacy and anonymity in the internet. We implement a new architecture to integrate multimedia data from the cloud using Activiti as a workflow engine and the sealed cloud IDGARD.

Copyrighted Content

From a user perspective, the access to files via the internet, for instance, by using mobile devices can be risky, because some information can be spied out by unauthorized persons. If some potential attackers retrieve user information while data is sent over the wire they may gain full access to the data storage. To prevent these attacks the usage of security tokens or passcodes (received via SMS) is necessary. IDGARD provides some of these additional security mechanisms.

From the provider perspective it is also difficult to secure data and protect it from unauthorized access or illegal sharing. Providers want to ensure that only one user or a specific user group is able to access data with the registered devices. By sending the unique keys to the registered devices, content providers can prevent the unauthorized sharing of data (e.g. by using a sealed cloud). There are similar examples in the nutrition domain where images or videos are already protected, for instance mycoachnutrition.com (MyCoachNutrition 2015). Like our approach, these services offer user individual content. In contrast to our work, they do not provide rapid scalability.

Definition of Workflows

Work is often organized as a sequence of individual tasks in which the progress can be observed (Hammer and Champy 1994). These individual tasks are linked to each other and they underlie a business objective or a policy goal (Workflow Management Coalition 1998). The automation of business processes is called workflow. According to the definition of the Workflow Management Coalition, a workflow is: "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." (Workflow Management Coalition 1998). In the remainder of this paper, we will use the term "workflow" as synonym for "business process". For our project, we model different cooking instructions using Activiti as a workflow engine. A simple workflow consists of a start event, a task with a data object and an end event (see Figure 1). A task or an activity describes a piece of work that forms a logical step in a process. To support the process execution the workflow activity requires human and/or machine resources for process execution (Workflow Management Coalition 1998). We use different tasks and data objects to describe a cooking control flow.

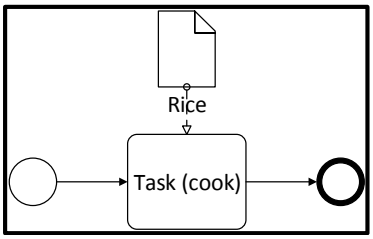

Figure 1: Sample workflow (own representation)

3 Architecture

General overview

As illustrated in Figure 2, the introduced architecture consists of three layers:

From the perspective of the user, the first layer or sub-component is based on the so called Ninja Web-Framework, which is required in order to handle the graphical interaction with the user by initiating a workflow instance.

Below the Web Framework, the Business Process Management (BPM) platform Activiti initially receives the user commands and controls the processing of the corresponding instance. The Activity engine requests relevant information from the data source (IDGARD) and hands them over to the user.

IDGARD, as the third or bottom layer of the architecture, is not just an external database but a sealed cloud solution. Therefore, it does not only store the data but provides API functions that ensure secure retrieval.

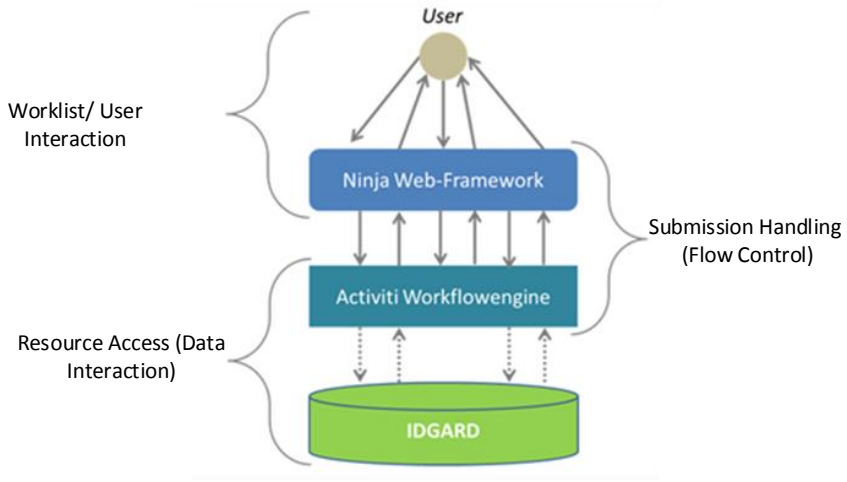

Figure 2: Architecture (own representation)

Reasons for Usage of a 3-Layer-Architecure and its Components

Since the goal is to develop and execute (cooking) workflows it is required to use Business Process Management and a solution to save the corresponding multimedia data. As a result, it is pre-determined to stage a 2-Layer-Architecure at minimum. Since it is a typical demand that users have not just local access to the workflows, a browser-based solution is reasonable. This kind of functionality is not offered by the already mentioned layers and therefore asks for an additional one.

Starting from the bottom layer, there is the complex question of how to save an application's data or where to put it. As Cloud Computing nowadays has already overcome just being a trend, it is appropriate and scientifically valuable to embed this idea into various contexts.

IDGARD´s sealed cloud is specifically designed to enhance security. Because protecting own content like graphically supported cooking steps or even whole recipes can be important, such a cloud service might be beneficial. This especially applies to companies that want to distribute such services via Internet and protect themselves against copyright infringements.

Concerning the choice of a workflow engine, there is a huge range of open source technologies available. In terms of basic functionality (e.g. integration in an IDE or graphically establishing workflows) they usually show similarities. To address our requirements properly we decided to use two different GUI's for our project. Activiti as the modelling GUI and the Ninja Web-Framework for the user worklist. However, Activiti convinces with a distinct manual, a large community and a clear, browser based testing environment.

To provide the worklist, the Ninja Web-Framework is useful because, as an integrated software stack, it already comprises many important libraries.

By using a MongoDB database, we save standardized keywords for each possible instruction and ingredient on the one hand, as well as the recipes with already assigned processing times, instructions and information on the other hand. We are planning to use time information for the retrieval of a process in our future work.

4 Implementation Concept

An abstract class in Java defines the fundamental set of functions, i.e. basic java members and abstract methods that deal with communicating information between java and JSON, and the log file for further checks and troubleshooting. Each function has still its own unique representation for requests and responses including the corresponding JSON formats. In order to successfully connect request- and response functionality for data interchange, some middleware classes are required. For providing cloud access, we have to send login-data and a random token of the client for identification purposes. The factory pattern is used to invoke requests and their corresponding response classes. The Java classes that realize the communication with the cloud server are directly implemented in the workflows, which we use for cooking instructions.

Workflow engines for Business Process Management are abounding. Examples for these are JBoss, jBPM or Activiti. For our workflow development, we use Activiti as an open source workflow engine. Activiti uses BPMN 2.0 as a modelling language and can be easily integrated with Java environments (Alfresco 2015). We used seven already created pasta cooking workflows from the work of Minor et al. and converted them into an Activiti workflow (see Figure 3 following Minor, Bergmann et al. 2010). Each cooking task is modelled by a service task. This kind of task enables us to invoke

a Java class for API cloud access. We also deposit the ingredients as data objects directly in the activity workflow. Other content like pictures or movies for cooking instructions are stored in the cloud.

In the example we have four individual instructions for the user. First, *cook* and *place in serving bowl* as well as *puree* should be conducted in parallel. Parallelization can be modelled using XOR-, AND- (symbolized by the plus), or LOOP-blocks (Schumacher, Minor et al. 2013). When both branches are finished the last task is *toss*. In each task, users get instructions with support of multimedia data from the cloud. To execute our designed workflows we use the Ninja Web framework. The Web framework (worklist) is a resource which performs the work presented by a workflow activity instance and

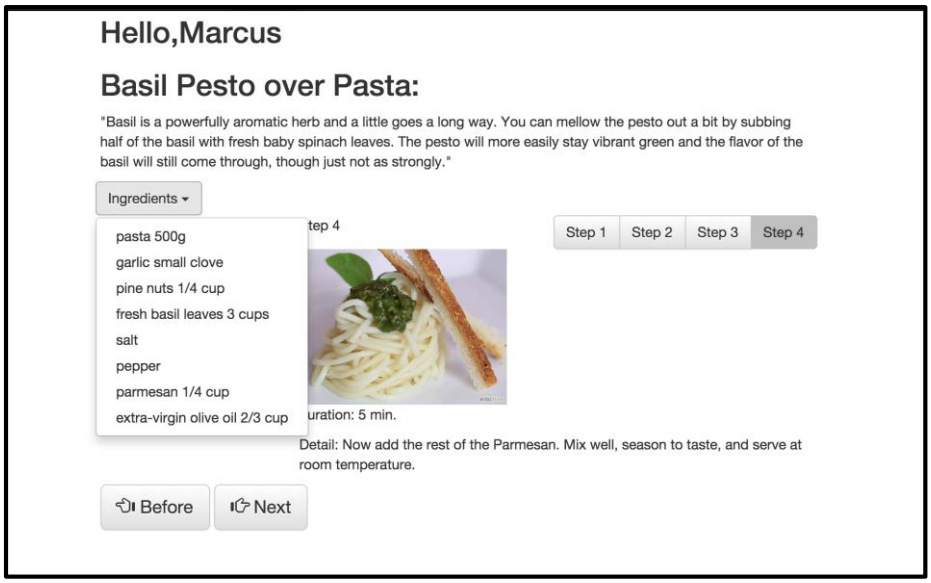

Figure 4: Current worklist (own representation)

therefore directly interacts with the user and supports her in executing her tasks (Workflow Management Coalition 1998). After successfully logging in, a user can choose to design her own recipe and upload new elements or choose ingredients to process. When searching for new processes for recipes in the database it is difficult to find a workflow or recipe which is well suited for the desired ingredients. To implement a search, case-based retrieval can be used. However, because of the small size of the recipe base in our solution this is not implemented yet.

Each recipe consists of several steps that are described by its linked ingredients, related instructions and actions that are supposed to be undertaken. In Figure 4, the prototypical implementation of our worklist is depicted. The sample picture is directly imported from the cloud. The non-multimedia information around the task is saved in a MongoDB database. When the user starts to cook she immediately sees which ingredients she needs, processing time, next steps and a video or picture to support her task. In the left top corner of Figure 4, one can see the ingredients to be used for the particular task. In the center is a management board with the instructions supported by a video or

a picture of the specific task. The workflow running in the background is modelled in BPMN 2.0. By pressing next step the user is guided to the next task of the related recipe as specified in the BPMN workflow. Thereby, we provide a worklist with different instructions for a kitchen chef.

5 Discussion and Conclusion

In our paper we present a novel approach to integrate a high security cloud storage (sealed cloud) in Business Process Management. We implement a new model for using a sealed cloud multimedia data storage for our workflow contents. The implementation of a case-based retrieval is not implemented yet. As a first step, we demonstrate a pasta-cooking workflow, which can be processed or uploaded by the user in a web form. The content is stored in the sealed cloud IDGARD of the Uniscon GmbH who provides the cloud infrastructure. The content and meta data is encrypted and protected from unauthorized provider or third-party access. By using cloud storage for our workflows we examine two main benefits: data protection and scalability. With the IDGARD solution third parties are not able to copy or even access the data.

Scalable storage provides the advantage of adapting storage up and down on-demand without using rare data space on physical disks of computers or other devices (Armbrust, Fox et al. 2009). By using IDGARD, we are able to scale our storage capacity in a flexible way in response to service usage and new customer demands. Since it is very difficult to estimate, what recipes and data with instructions will be added, scalability is very important. Apart from that, the evolution in terms of video quality has significantly risen in the past years (and probably keeps rising in the future) which contributes to the fact that scalability is an important factor because customers also tend to pressure companies to provide state-of-the-art content. The amount of data that goes along with this development is significant. It is also possible to migrate our workflow engine into a cloud solution in future. As a result, the whole platform can be scaled in response to service usage.

To sum up, our approach of using cloud storage for workflows in the cooking domain benefits from flexible scalability, higher privacy and data protection leading to a higher user experience especially for special interest groups of the cooking domain. The user no longer needs to use her limited, physical storage. She can store multimedia data with cooking instructions directly in the cloud. Our work also contributes to other future trends besides the cooking domain. Sealed cloud technology offers opportunities to use cloud storage without harming privacy or security regulations. This can be very important for audit companies who want to store audit-documents or other critical contents in the cloud.

7 Acknowledgment

We would like to acknowledge our partnership with the Uniscon GmbH who provided us with their sealed cloud service IDGARD.

8 References

Aalst, W. M., B. Benatallah and F. Casati (2003). Business Process Management, Springer-Verlag Berlin Heidelberg.

Alfresco. (2015). "Activiti mission statement." Retrieved 08.06.15, 2015, from http://activiti.org/vision.html.

Armbrust, M., A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica and M. Zaharia (2009). Above the Clouds: A Berkeley View of Cloud Computing, Technical Report, UC Berkeley Reliable Adaptive Distributed Systems Laboratory.

Ciovică, L., M. P. Cristescu and L. A. Fraţilă (2014). "Cloud Based Business Processes Orchestration." Procedia Economics and Finance **16**(0): 592-596.

Workflow Management Coalition (1998) "Terminology & Glossary." Retrieved 08.06.15, 2015, from http://www.wfmc.org/resources.

Hammer, M. and J. Champy (1994). Business Reengineering: Die Radikalkur für das Unternehmen. Campus-Verlag Frankfurt.

Minor, M., R. Bergmann, S. Görg and K. Walter (2010). Adaptation of cooking instructions following the workflow paradigm, In: ICCBR 2010 Workshop Proceedings, pp. 199-208.

MyCoachNutrition. (2015). "MyCoachNutrition." Retrieved 06.07.15, 2015, from http://www.mycoachnutrition.com/.

Rieken, R. (2015). Datensicherheit als Herausforderung im Cloud-Computing-Trend. In: Marktplätze im Umbruch**:** pp 751-759, Springer.

Santos, N., K. P. Gummadi and R. Rodrigues (2009). Towards trusted cloud computing. In: Proceedings of the 2009 conference on Hot topics in cloud computing. San Diego, California, USENIX Association.

Schulte, S., C. Janiesch, S. Venugopal, I. Weber and P. Hoenisch (2015). "Elastic Business Process Management: State of the art and open challenges for BPM in the cloud." Future Generation Computer Systems **46**(0): 36-50.

Schumacher, P., M. Minor and E. Schulte-Zurhausen (2013). Extracting and enriching workflows from text. In: Proceedings of the 2013 IEEE 14th International Conference on Information Reuse and Integration (IRI), pp. 285-292.

Wang, J., P. Korambath, I. Altintas, J. Davis and D. Crawl (2014). "Workflow as a Service in the Cloud: Architecture and Scheduling Algorithms." Procedia Computer Science **29**(0): 546-556.