Won't Take No for an Answer: Resource-driven Requirements Adaptation

Conference or Workshop Item

Won't Take No for an Answer: Resource-driven Requirements Adaptation

Amel Bennaceur*, Andrea Zisman*, Ciaran McCormick*, Danny Barthaud* and Bashar Nuseibeh*†
* The Open University, UK
† Lero - The Irish Software Research Centre, Ireland

***Abstract*—Adaptive composition dynamically and opportunistically uses and combines resources to best satisfy user requirements. However, when available resources cannot satisfy those requirements, no guidance or alternative options are offered by existing composition solutions. In this paper we address this issue by presenting an approach that tries to find substitutions for unavailable resources while satisfying the initial requirements. If no satisfactory substitutions are found, the requirements are adapted based on the resources available. Given that such requirements adaptation might be unbounded, we limit the search space guided by the available resources. Our approach ensures that alternative compositions given to users are achievable using available resources. We demonstrate the validity of our approach by implementing a prototype tool and applying it to support individuals in meal planning to reduce food waste.**

***Index Terms*—Requirements adaptation, dynamic composition, food waste**

I. Introduction

Requirements cannot always be satisfied using available resources. The notion of *satisficing* was introduced by Simon [1] to capture suboptimal, but best possible solutions, when optimal solutions cannot be achieved. It was also used in the context of analysing nonfunctional requirements [2]. This is important when dealing with resource-critical applications. Many real-world applications focus on resources such as budget constraints in project management or travel planning applications, energy constraints in smart cities and electricity grid applications, and availability of ingredients in food applications in family households. For example, when planning a family meal for which some ingredients are missing, one could use other ingredients or find recipes with available ingredients.

In this paper, we propose a *three way resource-driven adaptation* approach that (i) identifies resources that can be used to fulfil a given requirement; (ii) identifies alternative resources that are available and can be used as substitutes to fulfil requirements, or (iii) adapts requirements based on available resources. Instead of adapting requirements and then analysing how to design and implement those adapted requirements, the approach aims to best use available resources to *satisfice* requirements.

Several techniques have been proposed to support requirements adaptation due to changes and uncertainty inherent in self-adaptive systems [3]–[9]. RELAX [3] is a requirement language for self-adaptive systems that supports specification of non-invariant and variant requirements. FLAGS [4] is a goal model which defines countermeasures to be performed when one or more goals cannot be fulfilled. AutoRelax [6] is an approach that combines goal modelling [10] and requirement relaxation in order to support the analysis of tradeoffs [7], [8]. Awareness requirements [5] enable users to specify the extent to which other requirements should be satisfied at design time. At runtime, requirements are adapted accordingly based on environment conditions. SACRE [9] monitors and reacts to failure in requirements based on an *a priori* definition of possible requirements adaptations and runtime environment properties. Requirements adaptation techniques focus on specifying alternative requirements at design time to allow for uncertainty during execution. However, the specification of those alternative requirements may require knowing the resources available.

Dynamic composition approaches focus on making the best use of resources available to satisfy requirements, assuming a complete knowledge of the environment, but fail when requirements cannot be satisfied [11]–[18]. These approaches either identify compositions when they exist, or report that no composition can be identified, within a bounded solution domain. For systems with more than a few resources the search space can be large and complex. Search-based approaches aim to navigate the space to find an exact optimal solution [19]. However, when an optimal solution does not exist, users are not given guidance on how to revise their requirements.

Our approach aims to exploit the synergy between requirements adaptation and available resources. While techniques for requirements adaptation expand the solution space and search-based techniques prune the solution space looking for optimal solutions, our approach controls the expansion of the solution space guided by the search for nearly optimal solutions. Our contributions in this paper are:

- A *three way adaptation* technique that addresses the problem of unavailable resources and how to *satisfice* requirements by using alternative resources, or adapting the requirements. The approach first seeks a composition of resources that satisfy requirements. If this cannot be achieved, it looks for possible substitutions that offer alternative implementation of the requirements based on resources. As a last resort, it dynamically adapts requirements to offer users *satisfycing* options from which to choose, informed by the resources available.
- A *demonstrator in the food domain*. We apply our approach in the food domain in order to reduce food waste for individuals in households. We implemented a

prototype to illustrate and demonstrate the approach, and use it to evaluate the work in the domain of cooking.

The remainder of this paper is structured as follows. Section II presents the problem being tackled, introduces a meal planning running example used throughout the paper, and describes an overview of our three way adaptation approach. Section III details our adaptation approach. Section IV presents implementation of our tool and the experiments we conducted to validate the approach. Section V examines related work. Section VI concludes the paper and discusses future directions.

II. Motivation and Overview

Requirements adaptation is concerned with specifying alternative requirements for self-adaptive systems. Composition techniques target integration and interoperability and are often agnostic to requirements adaptation. In this section we discuss the differences and potential synergies between these two categories of techniques. We introduce an example to illustrate the need for combining these techniques and outline our three way adaptation approach.

A. Problem Statement

Requirements and compositions may not seem to fit together naturally as illustrated in Fig. 1. While requirements reside primarily in the problem space, compositions reside in the solution space. Requirements reflect an understanding of the environment, the needs of stakeholders, and the rationale behind the development of a proposed system; whereas compositions focus on the emergence of properties and behaviour by enabling individual resources to interact with one another. Moreover, requirements are often refined by decomposing a problem into smaller problems, while compositions aim to combine resources in order to support the implementation of more complex behaviour. The increasing deployment of mobile and ubiquitous computing technology is blurring the boundary between the problem and solution worlds [20]. Realising requirements through the composition of multiple existing resources is becoming a necessity. However, the question remains of *how to bridge the gap between requirements and compositions* in changing environments.

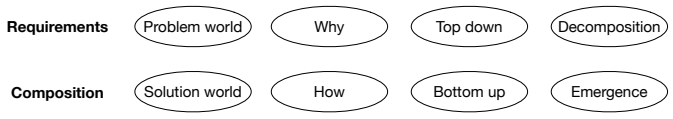

Fig. 1. Requirements vs Composition

Our aim is to integrate requirements-driven composition (with and without substitutions or resources) and resource-driven requirements adaptation. Therefore, we must answer the following questions:

- How to represent and evaluate the impact of substitutions on overall compositions?
- How to decide which requirements to adapt?
- When to switch from compositions to resource-driven requirements adaptation?

B. Motivating Example: Meal Planning

Food waste is an important problem in society. It is estimated that food waste per capita by consumers in Europe and North America is 95-115 kg/year [21]. Food waste is often caused by insufficient planning of purchases and consumption by individuals. Effective strategies to reduce wasteful behaviour should require minimum time and cognitive effort from consumers [22]. This is confirmed by initiatives such as IBM Chef Watson [23] and Foodie [24] that aim to assist users in their choices of recipes and ingredients. These solutions are dependent on users trying multiple ways to specify their requirements (recipes), instead of proposing recipes based on available resources.

The Feed me Feed me [25] exemplar describes an adaptive system based on the Internet of Things to support production, distribution, and consumption of food. In this paper, we use ideas and challenges from the Feed me Feed me exemplar [25] to focus on how our approach can support individuals in reducing food waste in households. Our approach starts with knowledge of the ingredient available in a kitchen, and relies on a knowledge base such as Cook's Thesaurus [26] to specify substitutions of specific ingredients.

In order to illustrate, let us consider the following simplified recipe: *GlutenFreeChocolateBrownie* = {*Chocolate, BrownSugar, AlmondFlour*}. Several cases can be considered:

❶ All ingredients of the recipe are available, in which case the user can proceed with the chosen recipe.

❷ Some ingredients of the recipe are available, others can be substituted using other available ingredients:

a) Multiple alternative options exist with some preferred over others. For example, *AlmondFlour* is unavailable, but can be replaced by either *HazelnutFlour* or *CornFlour* with a preference to *HazelnutFlour*. Both ingredients are available and the user can choose to proceed with the following list of ingredients {*Chocolate, BrownSugar, HazelnutFlour*}.

b) Some conflicts may arise due to the substitution. For example, *Chocolate* is unavailable, but can be replaced by *Cocoa* and *Milk*, both of which are available. However, milk must not be mixed with *NutFlour*, which includes both *AlmondFlour* and *HazelnutFlour*. As a result, the ingredients of the recipe will be {*Cocoa, Milk, BrownSugar, CornFlour*}.

❸ The necessary ingredients are unavailable and cannot be substituted using available ingredients. Other recipes similar to the first one, and using available ingredients are suggested. For example, both *Chocolate* and *Cocoa* are unavailable. Another recipe, close to the original one, but using only available ingredients can be proposed, such as *GlutenFreeBlondie*, which can be made using {*BrownSugar, HazelnutFlour*}.

C. Approach Overview

We formalise our approach using Jackson and Zave's framework for requirements engineering [27], which makes explicit

the relationships between requirements, specifications, and environment properties.

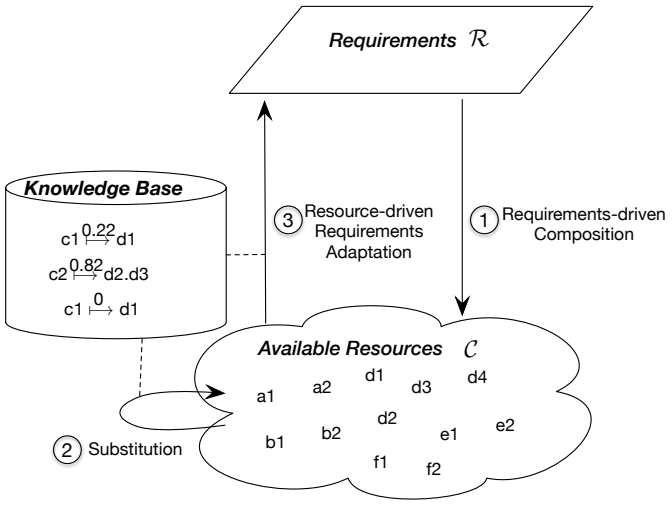

Fig. 2. Three Way Adaptation

The aim of our three way adaptation approach is to find a collection of resources that best satisfy requirements, or to adapt those requirements based on available resources. The first step seeks a set of resources that satisfy the requirements (see Fig. 2-①), which can be formalised as follows.

$$\text{Find } S \subseteq \mathcal{C}' \text{ such that } S \models \mathcal{R}.$$

where $\mathcal{R}$ is the set of requirements that we seek to satisfy and $\mathcal{C}'$ is the set of available resources. S represents the subset of available resources whose composition satisfies requirements $\mathcal{R}$. In the meal planning example, this corresponds to case ❶ where $\mathcal{R}$ = *GlutenFreeChocolateBrownie* and the user can proceed with the recipe as all ingredients are available.

It might be the case that no subset of the available resources can satisfy all requirements. We then seek to replace parts of the composition by semantically equivalent instances (see Fig. 2-②). The semantic equivalence is based on knowledge about possible matches and mismatches between resources, which can be formalised as follows:

$$\text{Find } S \subseteq \mathcal{C}' \text{ and } S' \subseteq \mathcal{C} \text{ such that } S \approx S' \text{ and } S' \models \mathcal{R}.$$

where $\mathcal{C}$ is the set of all resources and $\approx$ defines semantic equivalence between two sets of resources.

This semantic equivalence between two sets of requirements is obtained by finding a suitable match between each of their resources. S is the set of resources satisfying requirements $\mathcal{R}$ and S' is an equivalent set of available resources. In the meal planning example, this corresponds to case ❷ where some substitutes are available (*AlmondFlour* $\approx$ *HazelnutFlour* in case a) and (*Chocolate* $\approx$ *Cocoa.Milk* in case b), and the user can still proceed with the *GlutenFreeChocolateBrownie* recipe.

Where no suitable alternative is found, we relax the requirements according to the availability of resources (see Fig. 2-③).

$$\text{Seek } R' \approx_{\mathcal{R}} \mathcal{R} \text{ and } S \subseteq \mathcal{C}' \text{ such that } S \models R'.$$

where R' is the *closest* (as defined by the relationship $\approx_{\mathcal{R}}$) set of requirements that can be satisfied using available resources. In the meal planning example, this corresponds to case ❸ where R' = *GlutenFreeBlondie* and *GlutenFreeChocolateBrownie* $\approx_{\mathcal{R}}$ *GlutenFreeBlondie*.

III. Three Way Adaptation

This section details the steps of our three way adaptation approach. We describe how to achieve complete satisfaction of requirements using available resources. We also describe how to compensate for missing resources by substituting them with similar available resources, and the requirements adaptation step. This process is illustrated in Fig 3.

A. Requirements-driven Composition

In this first step we are driven by requirements: given a set of requirements and a set of available resources, the aim is to select the subset of resources that will satisfy those requirements. This problem can be formulated as a *Multi-Objective Constrained Optimisation Problem (MOCOP)*. A MOCOP is a tuple (X, D, T, U) where:

- $X = \{x_1, ..., x_n\}$ is a set of binary variables. Each available resource is associated with a variable with value 1 if the resource is selected, and value 0 otherwise.
- D is a function that associates each variable x_i with its domain $D(x_i) = \{0, 1\}$.
- $T = \{T_1, ..., T_m\}$ is the set of constraints. A constraint T_j is a mathematical relation defined over a subset $x^j = \{x^j_1, \ldots, x^j_{n^j}\} \subseteq X$ of variables, which restricts the values that the variables can take at the same time. The constraints can be domain specific or user specific. In the meal planning example, it is possible to have constraints forbidding the occurrence of two or more ingredients at the same time such as the use of *Milk* and *Lemon*, or constraints that are specific to a user such as excluding *Meat* for a vegetarian user.
- $U = \{U_1, ..., U_k\}$ is a set of objective functions whose values we seek to optimise. An objective function $U_{l=1..k}$ is defined over a subset of variables $Y \subseteq X$ and associates a utility—usually an integer or real value—to each assignment of Y.

Solving a constraint satisfaction problem consists of finding the tuple (or tuples) $v = (v_1, ..., v_n)$, where $v_i \in D(x_i)$ such that all constraints C_j are satisfied. Constraint programming (CP) is used to study combinatorial problems by stating constraints (conditions, qualities), which must be satisfied by the solution(s) [28]. It must be noted that the above definition of the CP problem and its solution techniques are suitable for variables of finite domains. Thus, CP uses constraints to state the problem in a declarative way, without specifying a computational procedure to enforce them. The latter task is carried out by a solver. The constraint solver implements intelligent search algorithms such as backtracking, branch, and bound, which are exponential in time in the worst case, but may be very efficient in practice. They also exploit the arithmetic properties of the operators used to express the

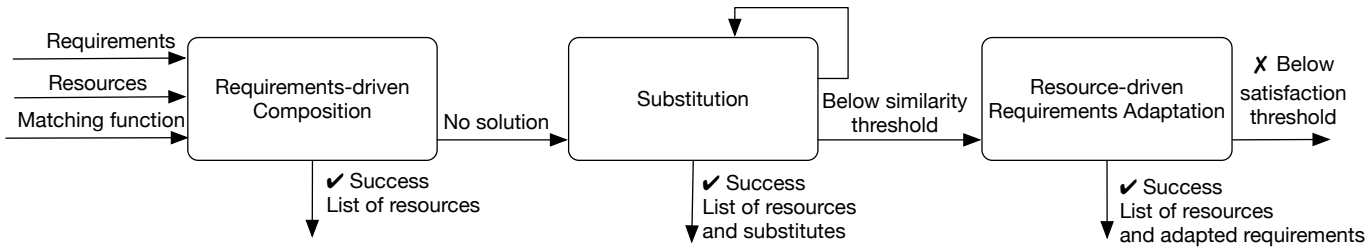

Fig. 3. From exact composition to requirements adaptation

constraint to quickly check solutions, to discredit partial solutions, and to substantially prune the search space.

Example. Let us consider the meal planning example described in Section II-B and its main elements.
Variables and Domains. Available resources are represented as a set of quantified ingredients available at home:

$$\mathcal{G}' = \{c' \mid c' = \{(c, q_c)\ ,\ c \in \mathcal{I}\}\}$$

where $\mathcal{I}$ denotes the set of ingredients and q_i is the quantity of the ingredient c. The first requirement is represented by a *recipe* r, which is also a list of quantified ingredients.

$$\mathcal{P} = \{r \mid r = \{(c, q_c)\ ,\ c \in \mathcal{I}\}\}$$

where each recipe specifies the quantities for one person and $\mathcal{P}$ is the set of all recipes.

Dietary preferences are captured by a function that measures how much each user likes a recipe. For simplicity we consider a number between 1 and 5, where 1 means that the user does not like the recipe and 5 means that the user likes it a lot. We also use value 0 to represent that a user cannot have the recipe (e.g., food intolerance, allergy, culture, or ideological preferences).

$$\mathit{Pref}:\ \mathcal{U} \times \mathcal{P} \rightarrow [0..5]$$

where $\mathcal{U}$ denotes the set of users.
Constraints. An example of a constraint is to avoid recipes that a user cannot have. Therefore, user preferences must be positive.

$$\forall u \in \mathcal{U}, \quad \mathit{Pref}(u, r) > 0$$

where r denotes the selected recipe.
Another constraint is to use only available ingredients:

$$\forall c \in r, \quad \mathit{Quantity}(c, r) \leqslant q_c$$

where $\mathit{Quantity}(c, r)$ denotes the quantity of the ingredient $c \in \mathcal{I}$ in the selected recipe r, and q_c denotes the quantity of the ingredient c in the groceries.
Objective function. An example of an objective function is to maximise user preferences.

$$max \sum_{u \in \mathcal{U}} \mathit{Pref}(u, r)$$

For example, assume we have three available ingredients {*Chocolate, BrownSugar, AlmondFlour*}. This is represented as a vector of three binary variables $[x_1, x_2, x_3] \in 2^3$ indicating whether an ingredient is selected. For the *GlutenFreeChocolateBrownie* recipe, all ingredients are available and must be used. The binary representation of the MOCOP solution is 111.

B. Substitution

When missing resources prevent the exact satisfaction of requirements, the next step is to attempt to replace parts of the composition by semantically equivalent available resources. The semantic equivalence is based on knowledge about possible matches and mismatches between resources. More specifically, we rely on a relation $M \subseteq 2^{\mathcal{C}} \times 2^{\mathcal{C}} \times \mathbb{Z}$ that specifies that a set of resources s_1 matches the set of resources s_2, with a degree of similarity a, noted $s_1 \overset{a}{\mapsto} s_2$. Note that $a = 0$ means mismatching sets. In the meal planning example, one could constraint mixing *Milk* and *Lemon* in recipes by specifying *Milk* $\overset{0}{\mapsto}$ *Lemon*. The use of degree of similarity offers a simple way to represent context and to forbid the simultaneous occurrence of some ingredients. The use of more elaborated context descriptions is an area for future work.

The aim is to find a composition S, that is similar to the requested composition and satisfies the requirements. We seek to maximise the similarity of the overall composition.

$$max \prod_{s' \in \mathcal{C}',\ s \in \mathcal{C}} M(s', s)$$

However, it is possible to have conflicts in the substitutions due to mismatches or constraints imposed by the requirements. In case of conflicts, it is necessary to backtrack the process in order to identify other possible substitutions that do not create conflicts. The approach also avoids *over substitution*; i.e. substitution of resources by other resources that are not compatible. An additional constraint is for the overall similarity between substitutions to be above a specified threshold:

$$\prod_{s' \in \mathcal{C}', s \in \mathcal{C}} M(s', s) > sim_{th}$$

where sim_{th} is the similarity threshold in which the substitution would not be satisfactory. This threshold can be specified by the user or set to any default positive value. As a result, the MOCOP needs to be updated by including an additional constraint and an objective function to maximise the similarity, and keep it above the specified threshold.

Example. Let us consider the meal planning example. Fig. 4 illustrates the database we use to represent the knowledge

base necessary to perform resource substitutions. Note the relationship between Ingredient and IngredientsList. In the case in which a substitute ingredient exists, the associated similarity is also specified.

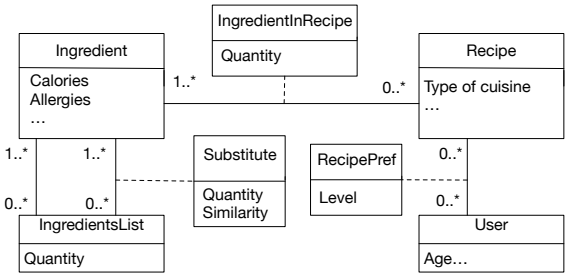

Fig. 4. Knowledge base for the meal planning example

Let us assume that the available ingredients are {*Chocolate, BrownSugar, HazelnutFlour, CornFlour*}. We recall that the original recipe was *GlutenFreeChocolateBrownie* = {*Chocolate, BrownSugar, AlmondFlour*}. The knowledge base specifies the following substitutions. *AlmondFlour* $\overset{0.8}{\mapsto}$ *HazelnutFlour* and *AlmondFlour* $\overset{0.4}{\mapsto}$ *CornFlour*. Given that the aim is to maximise similarity, the proposed recipe with substitution will be *GlutenFreeChocolateBrownie* = {*Chocolate, BrownSugar, HazelnutFlour*}, with a similarity level of 0.8.

Let us now assume that the available ingredients are {*Cocoa, Milk, BrownSugar, HazelnutFlour, CornFlour*}, while the knowledge base specifies two additional matchings: *Chocolate* $\overset{0.4}{\mapsto}$ *Cocoa, Milk* and *HazelnutFlour* $\overset{0}{\mapsto}$ *Milk*, which designates that the two ingredients cannot be mixed together. In this case, the proposed recipe with substitution will be *GlutenFreeChocolateBrownie* = {*Cocoa, Milk BrownSugar, CornFlour*} with a similarity of 0.32 (= 0.8 × 0.4). However, when similarity drops below a certain threshold, it might be better to switch to another recipe, as discussed below.

C. Resource-driven Requirements Adaptation

When available resources cannot satisfy requirements or provide acceptable alternatives/substitutions, our approach adapts the requirements based on available resources. The user chooses either to accept the suggested requirements adaptation or to increase the available resources.

One issue when adapting requirements is the explosion in possibilities. In our work, rather than adapting the requirements and restarting the search for satisfaction of the new requirements, the approach identifies configurations that best fit the requirements given the resources at hand. The availability of resources helps reduce the search space and performs a smart exploration during requirements adaptation. It is not only about minimising resources, but also about choosing alternative requirements and optimising preferences (i.e., constraints and priority associated with those constraints). Resources help requirements adaptation in two ways (see Fig. 5). First, resource constraints allow us to filter out unfeasible requirements. Second, they guide the space exploration by selecting those requirements whose satisfaction requires similar resources to the initial requirement.

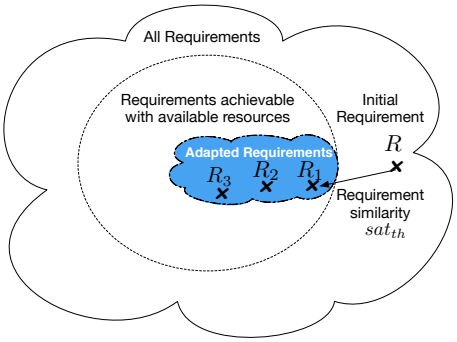

Fig. 5. Resource-driven requirements adaptation

The confidence in the requirements adaptation can be measured through the proportion of shared resources or domain specific measures. In the meal planning example, confidence can be measured using the ratio of shared ingredients between recipes. Other solutions can include common flavour combinations, co-occurrence of ingredients as in IBM Chef Watson [23], or health benefits as in Yum-me [29].

Users may also be able to shape the decision and adaptation process by choosing the appropriate options and giving feedback. In this case, the computation of similarity between requirements can build on the large body of work in recommender systems [30]. The main goal of relying on user feedback is to converge towards the best alternatives for satisficing user requirements over time and compensating for requirements uncertainty [31] [32]. Requirements adaptation is improved as our knowledge about the world is improved.

The algorithm for the three way adaptation approach (Algorithm 1) is described below. It takes as input a set of requirements, available resources, and a matching relation. It embeds two parameters: sim_{th} for specifying acceptable overall similarity for substitution, and sat_{th} for specifying acceptable degrees of similarity between requirements.

The first step is to construct a MOCOP with variables indicating whether a resource is selected (see Section III-A), and to transform functional requirements into constraints and non-functional quantitative requirements into objective functions (Lines 1-5). If a solution to MOCOP exists, then there is a selection of resources that exactly satisfy the given requirements (Lines 6-9). Otherwise, it tries to execute substitutions.

The second step is to substitute some of the missing resources so that the similarity is maximised and no known conflicts exist (see Section III-B) (Lines 10-11). If a solution exists, then there is a satisfactory selection of resources that meet the original requirements (Lines 12-15).

The third step is to adapt the requirements by selecting those that can be satisfied using available resources. It calculates the set of available resources that are considered in $\mathcal{R}$ and computes the corresponding partially ordered power set (Line 16). For each set T, within this power set, it computes potential requirements that involve those resources and have the remaining resources available (Line 19). The $Stop$ parameter is a condition for interrupting this computation. Given that the search space for finite feasible requirements can be very

ALGORITHM 1: Three Way Adaptation

Input: Requirements ($\mathcal{R}$), Resources ($\mathcal{C}$), Available resources ($\mathcal{C}'$), Matching Relation M
Output: Subset of available resources to compose (S), Adapted requirements (R')

1 Build MOCOP $A = (X, D, T, U)$
2 $X = \{x_i, 1 \leqslant i \leqslant |\mathcal{C}'|\}$
3 $D(X) = \{1, 0\}^{|\mathcal{C}'|}$
4 $T = Req2Constraints(\mathcal{R})$
5 $U = Req2Objectives(\mathcal{R})$
6 $S = SolveMOCOP(A)$
7 **if** $S \neq \emptyset$ **then**
8 | **return** S
9 **end**
10 $T = T \cup \left(\prod_{1 \leqslant i \leqslant |\mathcal{C}'|}^{|\mathcal{C}'|} M(x_i.\mathcal{C}', \mathcal{C}) \leqslant sim_{th} \right)$
11 $U = U \cup \left(\prod_{1 \leqslant i \leqslant |\mathcal{C}'|}^{|\mathcal{C}'|} M(x_i.\mathcal{C}', \mathcal{C}) \right)$
12 $S = SolveMOCOP(A)$
13 **if** $S \neq \emptyset$ **then**
14 | **return** S
15 **end**
16 *SharedAvailResOrdPowerSet* = $2^{Resources(\mathcal{R} \cap \mathcal{C}')}$
17 **while** *SharedAvailResOrdPowerSet*$\neq \emptyset$ **do**
18 | T =*Select*(*SharedAvailResOrdPowerSet*)
19 | *PotentialReq* = *FilterReq*($Stop, \mathcal{C}'$,T)
20 | **if** $R' \neq \emptyset$ **then**
21 | | $R' = SelectFirstRequirements(\mathcal{C}')$
22 | | $sat_{th} = |T| \div |Resources|$
23 | | **return** R'
24 | **end**
25 | *SharedAvailResOrdPowerSet* = *SharedAvailResOrdPowerSet* - T
26 **end**
27 **return** *Fail*

big, the search needs to be bounded either by time or by the number of explored solutions.

One of these requirements is selected if such a requirement exists (Lines 20-24). The selection may require domain specific information. For example, the preferred recipe for a user. The confidence in the adaptation is calculated as the ratio between the number of initial ingredients in the original requirement and the number of the adapted requirements. A more elaborate calculation can include preferences. The algorithm fails (Line 27) if all sets in the power set are explored (Lines 17-26) and a suitable adapted requirement is not identified.

Example. Let us consider the meal planning example. The similarity between recipes is calculated as the ratio of common ingredients in the recipes and all ingredients in the original recipe. Let us assume that the available ingredients are: {*BrownSugar, AlmondFlour, CornFlour, Orange*}. The necessary ingredients are unavailable and cannot be substituted using available ingredients. Other recipes similar to the first one and using available ingredients are suggested. For example, *GlutenFreeBlondie*, which uses {*BrownSugar, AlmondFlour*}, with a sat_{th} of 0.67; and *OrangeCake*, which uses {*BrownSugar, CornFlour, Orange*}, with a sat_{th} of 0.33.

The approach also supports constraints under specific resources. The user can specify ingredients that need to exist in a recipe. For example, if the user provides a constraint on the *AlmondFlour* ingredient, then the *OrangeCake* recipe would not be selected since it does not contain *AlmondFlour*. While recommendation of recipes based on the number of shared ingredients is simple, other measures can include user preferences and behaviour, building on past interactions and feedback of the users. This is an area for future work.

IV. Evaluation

We developed a prototype tool R2A (Resource-driven Requirements Adaptation) and used it to experiment with the meal planning case study. We first examine theoretical aspects of the three way adaptation approach and its complexity. We then report the results of experiments using R2A for the meal planning with different sizes of real-world knowledge base containing up to 1000 recipes and over 36000 substitutions. Finally, we discuss the limitations and possible enhancements of our approach. The evaluation covers the following properties of our approach:

- *Feasibility.* We provide tool support for the three way adaptation and evaluate it with scenarios of varying complexity in the domain of meal planning.
- *Performance.* We measure the time to perform each step in the three way adaptation to show that our approach can be applied at runtime in practical cases.
- *Scalability.* We show that by driving selection by both resources and requirements we are able to deal with a high number of ingredients and recipes in the case of meal planning, while keeping good performance time.
- *User acceptance.* We present the results of a preliminary usability study for meal planning. This study aims to evaluate features that are most useful for users.

A. Complexity

Composition with substitutions is NP-complete and can be proven using polynomial-time reductions from the knapsack problem [33]. We recall that in the knapsack problem, we are given a set of n elements, each of which with a size $s_1, \ldots, s_n$ and a value $v_1, \ldots, v_n$. We are also given a capacity B and value V and must seek a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} s_i \leq B$ and $\sum_{i \in S} v_i \geq V$.

The first step is to transform an instance of the knapsack problem into an instance of composition with substitutions. Each element of the knapsack problem represents a resource. We also have one additional element $n + 1$ representing the only resource that would satisfy the original requirement, but that is not available. The value of each element $v_{1 \leq i \leq n}$ represents the similarity between elements $n + 1$ and i. The size of each element $s_{1 \leq i \leq n}$ represents a quality attribute of the resource (e.g., quantity), and a quantitative requirement is to minimise this quality considering a given threshold B. To

get a solution to the knapsack problem, it suffices to pick the subset of resources that maximises similarity and keep it above the threshold $sim_{th} = V$, as well as satisfies a minimisation requirement with a threshold B. Therefore, composition with substitution is NP-hard. Further, we can compute the similarity between the original requirements and one with substitution in polynomial time. Hence, we can state that computation of substitutions (alternative composition) is NP-complete.

As some requirement might have an infinite set of values (e.g., quantitative requirements with rational or real values), there might be an infinite set of possible requirements adaptation. The approach uses heuristics based on similarity with the original requirements. The set of potential adaptation of requirements is bounded, in order to guarantee that the algorithm terminates.

B. Implementation and Empirical evaluation

In order to validate our approach, we implemented the R2A tool to perform the three way adaptation automatically. R2A is implemented in Java and is available at `https://github.com/amelBennaceur/R2A`. It takes as input a set of requirements, a set of resources, and the path to a database describing the similarity relation. It returns either the set of resources available or substitutions that satisfy the original requirements, or an ordered set of possible adapted requirements.

We ran the experiments on a MacBook Pro laptop with 2.8 GHz Intel Core i7 processor and 8 GB memory, and configured the heap memory of the JVM to the maximum. We evaluated the performance of R2A with the example scenario used in this paper. The results are presented in Table I and show that the tool provides the desired outcomes in short times.

We evaluated the approach in terms of performance and accuracy. In this evaluation, we used real-world data with variations in the size of the databases, the complexity of recipes, and the similarity between ingredients.

TABLE I
PROCESSING TIME (IN MILLISECONDS) FOR EACH CASE OF THE MEAL PLANNING EXAMPLE

Case	Description	Time
❶	Requirements-driven composition	1
❷ - a)	Substitution	1
❷ - b)	Substitution and dependency	2
❸	Resource-driven requirements adaptation	65

In the evaluation, the data set consists of three databases with different sizes: DB_1 contains 100 recipes, DB_2 has 500 recipes, and DB_3 contains 1000 recipes. These are real recipes extracted from Epicurious [34]. In the evaluation we also use 36000 substitutions extracted from BHG [35].

We considered three recipes of different sizes in order to evaluate a different number of substitutions. The recipes are:

- R_1: *Thai Summer Roll* - with five ingredients;
- R_2: *Banana Layer Cake with Cream Cheese Frosting* - with 11 ingredients; and
- R_3: *Shawarma-Spiced Chicken Pita with Tahini-Yogurt Sauce* - with 20 ingredients.

We selected these recipes in order to be able to test the approach for a high number of substitutes.

Performance evaluation for requirements-driven composition. The first set of experiments aim to demonstrate that the approach works independently of the number of ingredients and recipes. For each case, we made sure that all the required ingredients were available. The time to compute requirements-driven composition for all the three databases and three recipes is quasi-instantaneous (less than 2 milliseconds).

Performance and accuracy for composition with substitution. The second set of experiments focuses on substitutions. For each case, some ingredients in the recipes were unavailable. We varied the ratio of unavailable ingredients as 30%, 50%, and 80% for each case. The results are shown in Table II. One can note that even when the number of necessary substitutions increases, the processing times remain low. This is partly because we pre-compute substitutions into a hash map, which facilitates retrieval. As the number of missing ingredients increases, the similarity drops to near 0.

TABLE II
PROCESSING TIME IN MILLISECONDS AND (SIMILARITY) FOR COMPOSITION WITH SUBSTITUTIONS ACCORDING TO THE SIZE OF THE DATABASE AND THE RECIPES AND NUMBER OF MISSING INGREDIENTS

Case		R_1	R_2	R_3
DB_1	30%	1 (0.41)	1 (0.2)	2 (0.2)
	50%	1 (0.32)	1 (0.1)	3 (0.02)
	80%	1 (0.25)	1 (0.03)	8 (6.7E-4)
DB_2	30%	1 (0.41)	1 (0.2)	3 (0.2)
	50%	1 (0.32)	1 (0.1)	6 (0.02)
	80%	1 (0.25)	1 (0.03)	8 (6.7E-4)
DB_3	30%	1 (0.41)	1 (0.2)	3 (0.2)
	50%	1 (0.32)	2 (0.1)	7 (0.02)
	80%	1 (0.25)	2 (0.03)	9 (6.7E-4)

Performance and accuracy for resource-driven requirements adaptation. The third set of experiments considers requirements adaptation. In this case, we made sure that some ingredients are missing and have no substitution available. Similarly to the previous experiment, we varied the number of missing ingredients from 30%, to 50%, to 80%. Table III presents the processing time and the level of similarity between the original recipe and the first suggested recipe.

Note that the processing time for some of the 80% cases is lower that that for 50%, this is due to the fact that for a larger number of missing ingredients, several potential candidate recipes are ignored due to unavailable ingredients. Another aspect is due to the similarity calculation used in the approach, which is based on the number of shared ingredients between

recipes. For example, if we do not set constraints about necessary ingredients for R_2: *Banana Layer Cake with Cream Cheese Frosting*, the second recommended recipe is *Aztec Chicken*, as they both use *flour* and *buttermilk*. However, the first one is a cake while the latter is a savoury dish. Therefore, one must specify the ingredients that are key for a certain recipe, or use a more refined similarity function.

TABLE III
PROCESSING TIME IN MILLISECONDS AND (SIMILARITY sat_{th}) FOR ALTERNATIVE COMPOSITION ACCORDING TO THE SIZE OF THE DATABASE AND THE RECIPES AND NUMBER OF MISSING INGREDIENTS

Case		R_1	R_2	R_3
DB_1	30%	260 (0.2)	174 (0.09)	203 (0.1)
	50%	181 (0.2)	313 (0.09)	119 (0.05)
	80%	128 (0)	122 (0)	90 (0)
DB_2	30%	242 (0.2)	180 (0.27)	138 (0.15)
	50%	119 (0.2)	235 (0.18)	184 (0.1)
	80%	116 (0.2)	171 (0.18)	133 (0.05)
DB_3	30%	191 (0.4)	176 (0.27)	358 (0.15)
	50%	172 (0.2)	215 (0.18)	290 (0.1)
	80%	153 (0.2)	267 (0.18)	190 (0.05)

C. Preliminary User Evaluation

In order to analyse the acceptability/usability of our approach, we developed an online survey where we asked participants which alternatives for recipes R_1, R_2, and R_3 they would choose, and the reasons for which the participants would not accept the alternatives, if any. In the following we briefly describe the study and summarise our findings.

Methodology. We developed an anonymous survey using Google Docs and advertised it through emails and social networks. The survey and its results are available at `https://bit.ly/2SbbPGG`. Similar to the performance evaluation, for each recipe, the survey provided a list of substitutions corresponding to different ratios of unavailable ingredients (30%, 50%, and 80%). The survey also presented three potential swaps for each recipe and asked users to indicate reasons for not choosing some alternative ingredients or swaps. It also asked if users would rely on such tool, and which features they would like to be implemented.

We recruited 24 participants with different levels of self-reported cooking experience: 17 people had average cooking experience, five people had none or little cooking experience, and two people were expert cooks.

Results. The results of the survey showed that 92% of participants would choose a substitution or a swap for R_1 (starter) or R_3 (main). This number drops to 79% for R_2 (dessert/cake). The reason for this was due to the fact that people did not trust substitutions for baking dishes, stating that: "*When it comes to cakes I always abide by the recipe.*" or "*Banana Layer Cake is very specific. Changing anything changes the entire recipe*".

The results demonstrated that the notion of trust is important when dealing with automated suggestions. Participants declared that they will not swap recipes when "*Not familiar enough with recipe*". The fact that they could not imagine the flavour was also mentioned: "*I would try to imagine what type of taste and flavour the recipe has*". Participants would often swap a recipe if the main ingredient was missing, or if there were too many substitutions with appropriate levels of substitutions varying from one participant to another. Participants would also swap a recipe if they have time constraints, for example: "*When I don't have the time to go to the supermarket and some friends are coming for dinner*".

The usability of the tool varied depending on the level of the participants' expertise. 50% of the participants said that they would use such a tool in their everyday cooking. All expert cooks did not want to use the tool as they assumed that they have the skills to do it themselves declaring: "*You could easily take what you have to hand and adapt*", "*I am quite good in thinking about alternative ingredients as I cook everyday"*, or "*the tool would help many out there, however the question was directed at someone who is both a capable cook - they have also learned to adapt recipes*". The average cooks who did not want to use the tool mentioned time and ease of use as reasons: "*When I am in a hurry, I don't have time to use a tool*" or "*Seems like too much work*". Participants with little or no experience agreed to use the tool.

Findings and Next steps. The results of the survey also showed that the need for substitutions and requirements adaptation when dealing with recipe planning depends on expertise and confidence of the participants. The tool would be most useful for those who do not know how to adapt/change a recipe themselves. One incentive for expert cooks could be to offer a platform to share their expertise with other users and, therefore, improve the knowledge base and possible substitutions and swaps of recipes.

The context in which substitutions or recipe swaps are made is also important. Some participants would accept alternatives when preparing meals for friends or for certain types of dishes, but not for baking. The requirements adaptation needs also to consider the nature of the requirement (e.g., a main ingredient, which can be modelled as a hard constraint). Flavour is a difficult concept to represent and reason about automatically, especially as it is a personal property. More data is required to evaluate accuracy in flavour combinations.

While this first prototype focused on the functionality of the tool, the interface and ease of use is key in its adoption. More use of the tool may also lead to an increase in trusting substitutions and swaps by the users. Other incentives such as reducing food waste may encourage users who value sustainability (both environmental and economical). As future work we plan to evaluate the tool for an extended period and analyse its benefit for food waste.

D. Discussion

Our initial evaluation demonstrated that the three way adaptation is feasible and can propose substitution of ingredients,

or identify new requirements based on available resources, for different sizes of databases, different types of recipes, and distinct numbers and types of unavailable ingredients. The usability experiment also demonstrated some promising results for acceptance. We made several simplifying assumptions in order to implement and empirically evaluate our approach. We discuss how some of these assumptions can be relaxed.

Context. We assumed that substitutions are not context dependent. For example, an ingredient substitution can be applied to any recipe as long as it does not mismatch (conflict) with other ingredients. However, a substitution and its related similarity may depend on the style or other characteristics of the recipe. IBM Chef Watson computes the likelihood of how well ingredients can be mixed together. In contrast, our aim is not to suggest recipes in a creative way (i.e. generating or learning new knowledge), but rather to find a recipe, or variations of a recipe, that best match a desired dish and available ingredients (resources). The aim in our approach is to find the best recipe using existing knowledge to match available ingredients. Our approach can be extended to consider user feedback to select combinations of ingredients, or to enrich the shared knowledge (database) of recipes and substitutions. Moreover, the work can also be extended to associate substitutions and similarities among resources with the context in which they are employed.

Resources can also be time dependent. We can consider components or services with an elaborate behaviour where composition is not only related to the selection, but also to the coordination of behaviours or their mediation [12]–[18]. We aim to build on our previous work on services composition and mediation to support behaviour during requirement adaptation.

Qualities and tradeoffs. The work in this paper focuses on the relationship between resources and requirements, without considering conflicts and tradeoffs between qualitative requirements. The work in [36] specifies how to relate requirements and features using KAOS goal modelling, and uses a composition to realise the adaptation. However, this work is driven only by requirements. The DeSiRE [37] approach proposes a quantitative measure for satisfaction of non-functional requirements in order to explore trade-offs between non-functional requirements. A possible area for future work is to explore how composition can drive tradeoffs.

The work presented in this paper does not support adapting requirements while performing substitution. Quantifying adaptation may allow the use of constraint solving in requirements adaptation and, therefore, combining adaptation with substitution. Existing work on Weighted Constraint Satisfaction Problems [38] may provide other efficient solutions. Furthermore, the approach in this paper relies on explicit numerical values associated with each substitution. In [39] the authors propose a partial valuation structure approach for both quantitative and qualitative constraints. We intend to investigate the use of this approach to deal with cases where one can partially rank the constraints without using specific values.

Generalisation. Three way adaptation seeks to optimise the use of available resources to satisfice requirements. It aims to find the appropriate tradeoff between what users want (requirements), what they have (resources), and what is known (knowledge base). It can be applied to resource-critical systems and domains that seek to make the best use of available resources and can tolerate some degradation or deviation of the requirements. A travel agency example used to illustrate composition [40] as a holiday package consists of a hotel, a flight, and a car. When the requirements cannot be satisfied, several possible adaptations can be made to assist the user in making their decision, rather than letting the users test and see. Another illustration is in the classic example of the Meeting Scheduler [41], which can also benefit from requirements adaptation, as proposed in our work, in order to change rooms or even relax the constraints in meeting dates.

We plan to evaluate our approach in terms of behavioural change, continuous-time, and qualitative constraints using other examples such as budget constraints in project development or holiday trip applications, energy constraints in smart cities, and cyber-physical-social systems. We also intend to conduct empirical studies in order to evaluate relevance of the suggested adapted requirements, and compare the suggested requirements with intuitive changes proposed by the users, as well as evaluate how to learn and update the threshold for similarity and satisfaction. We are considering the use of the approach to support specific diet and health restrictions, as well as planning meals for an extended period (e.g., a week or a month), and adapting the meals depending on how they were followed by the users.

Threats to Validity. There are both internal and external threats to validity in our evaluation. An internal threat is related to the similarity levels between ingredients used in the evaluation, which were specified by the authors. However, these values were specified based on information available in a real cook thesaurus, Better Homes and Garden (BHG) [35]. We plan to use methods for qualitative constraints analysis to deal with cases where substitutions cannot be associated with specific valuesbut can be partially ordered.

With respect to external threat to validity, the work was evaluated in the meal planning domain. Therefore, the results may not be generalisable. However, the data used in the evaluation was from real cook thesaurus and databases (Epicurious [34]), BHG, IBM Chef Watson [23]). We intend to evaluate the work in other resource-critical domains such as budget constraints in project development or holiday trip applications and energy constraints in smart cities.

V. Related Work

Our three way adaptation approach focuses on applications involving 'static' resources and do not consider behavioural specification of software components. Many real-world applications focus on such resources including but, not limited to, energy, food ingredient, or calendar scheduling. Our approach is related to existing research in a number of areas, in particular, (1) requirements adaptation, (2) dynamic composition, and (3) food-based applications. We summarise some of the most relevant work in the following.

A number of proposals have been made to support description of **requirements for adaptation**, including: (i) *requirements reflection* which treats the requirement model as an executable model whose states reflect the state of the running software [42]; (ii) *awareness-requirements* which considers the extent to which other requirements should be satisfied [5], [9]; (iii) *numerical quantification* of requirements so that their parameters can be estimated and updated at runtime [43]; (iv) *quantification of requirements satisfaction* [37], and (v) *rewriting of requirements* to support uncertainty in the environment so that requirements are not violated when the system encounters unexpected conditions [3].

Requirements also need to be adapted when dealing with self-adaptive systems. In these cases, it is necessary to monitor whether the systems satisfy or violate requirements [44], often by means of a feedback loop in the environment [45]. This leads to questions of which parts of the environment need to be monitored, how to monitor them, and how to infer requirement satisfaction from the recorded data. In addition, self-adaptive systems have to relate their requirements to the system architecture at runtime (i) by switching the behaviour in response to monitored changes in the environment [46]; (ii) by exploiting a layered architecture in order to identify alternative ways of achieving goals when obstacles are encountered; and (iii) by reconfiguring components within the system architecture [47].

The work in [48] supports different specifications to be configured in order to satisfy predefined requirements. This solution requires knowledge of all potential sets of requirements and associated specifications, rather than automatically reacting to changes in the environment. Controller synthesis can also be used to generate software that satisfies requirements at runtime through decomposition and assignment of multiple agents [49], or by composing existing software components.

In [32] the authors emphasise the need for social adaptation, where the software system analyses users' feedback and updates its behaviour to best satisfy the requirements. With the prevalence of mobile and ubiquitous technology, it is becoming easier to have a better understanding of user preferences, and one can aim to compose both digital and social services [50]. These techniques complement ours as they may be used to calculate similarities between requirements and drive their adaptation. Approaches that support qualitative specification of constraints [39] could be used to deal with cases where explicit values cannot be assigned to substitutions.

Dynamic composition has been the subject of a large body of work. Looking for an optimal composition that satisfies multiple requirements is challenging. For systems with more than a few components the search space is unmanageable, large and complex. In [40] the authors propose to select a concrete service for each abstract service defined within a service composition, in order to optimise overall quality of service. However, this solution does not enable reasoning about requirements at runtime. There are a large body of work, especially in the services domain, that focuses on requirements-driven composition. A complete survey of these approaches is beyond the scope of this paper and we refer the interested user to [11] [51]. Our work focuses in the interplay between requirements and resources.

Recently, we have been experiencing the development of **food-based applications**. In ColibriCook [52], machine learning and user feedback are used to compute similarities between ingredients, store them in an ontology, and use them within recipes. The main objective of the approach is to generate knowledge of substitutions. This approach is complementary to ours. Overall, ontologies have been extensively used in the food domain. However, these methods often require a complete set of relations between recipes and their adaptations.

IBM Chef Watson [23] allows the user to choose an ingredient, a dish or a style of cuisine. It then presents four ingredients that work well together and allow the user to change any of those by selecting one of several suggested alternatives. The application presents a selection of recipes that use the selected ingredients based on existing recipes. However, the recipes may contain ingredients that are not available, in which case no alternative recipes are suggested. Furthermore, the above techniques consist of an ad-hoc solution for meal planning with no generalisable techniques.

Foodie [24] is a conversational agent that uses IBM Watson technology to provide recipes for cooks, taking into consideration dietary needs, constraints, cultural preferences, and even available ingredients in a house. Similar to Foodie, our work can also help individuals make the best use of the ingredients they have at home in order to realise a recipe of their choice. Our work uses family meal planning as an illustrative example and is applicable to support resource-driven satisficing of requirements in different domains (e.g., budget constraints, electricity consumption). Furthermore, Foodie is a conversational recommender system, while our work is a three way adaptation approach that supports resource substitutions and requirements adaptation based on available resources.

VI. Conclusion

In this paper we presented a three way adaptation approach to support satisficing requirements when resources are not available. Our approach first tries to identify a composition of resources that satisfy requirements. If unsuccessful, the approach searches for possible resource substitutions that can be used to satisfy the requirements. If these substitutions are unsatisfactory, the approach adapts the requirements based on availability of resources and suggests new requirements. We evaluated the approach to support meal planning in order to reduce food waste in family households.

We are investigating the use of the approach in other domains like smart cities and Internet of Things. For the meal planning and food waste domains, we are considering the use of the approach to support specific diet and health restrictions, as well as planning meals for an extended period based on available ingredients. We are also extending the approach to support goal modelling techniques for compositions to drive balancing trade off among requirements and conflict resolution. Another extension of the work is to allow for coordination of component behaviour within a composition.

ACKNOWLEDGMENT

The authors would like to thank Daniel Gooch for his valuable feedback and help with the use study. We acknowledge SFI grant 13/RC/2094 and EPSRC support.

REFERENCES

[1] H. A. Simon, "Rational choice and the structure of the environment," *Psychological Review*, vol. 63, no. 2, pp. 129–138, 1956.

[2] J. Mylopoulos, L. Chung, and B. A. Nixon, "Representing and using nonfunctional requirements: A process-oriented approach," *IEEE Trans. Software Eng.*, vol. 18, no. 6, pp. 483–497, 1992. [Online]. Available: https://doi.org/10.1109/32.142871

[3] J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, and J. Bruel, "RELAX: a language to address uncertainty in self-adaptive systems requirement," *Requir. Eng.*, vol. 15, no. 2, pp. 177–196, 2010. [Online]. Available: https://doi.org/10.1007/s00766-010-0101-0

[4] L. Baresi, L. Pasquale, and P. Spoletini, "Fuzzy goals for requirements-driven adaptation," in *Proceedings of the 2010 18th IEEE International Requirements Engineering Conference*, ser. RE '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 125–134. [Online]. Available: http://dx.doi.org/10.1109/RE.2010.25

[5] V. E. S. Souza, A. Lapouchnian, W. N. Robinson, and J. Mylopoulos, "Awareness requirements for adaptive systems," in *Proc. of the Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS@ICSE*, 2011, pp. 60–69. [Online]. Available: http://doi.acm.org/10.1145/1988008.1988018

[6] E. M. Fredericks, B. DeVries, and B. H. C. Cheng, "Autorelax: automatically relaxing a goal model to address uncertainty," *Empirical Software Engineering*, vol. 19, no. 5, pp. 1466–1501, 2014. [Online]. Available: https://doi.org/10.1007/s10664-014-9305-0

[7] E. Letier and A. van Lamsweerde, "Reasoning about partial goal satisfaction for requirements and design engineering," in *Proc. of the 12th International Symposium on Foundations of Software Engineering, FSE*, 2004, pp. 53–62. [Online]. Available: http://doi.acm.org/10.1145/1029894.1029905

[8] L. Pasquale, P. Spoletini, M. Salehie, L. Cavallaro, and B. Nuseibeh, "Automating trade-off analysis of security requirements," *Requir. Eng.*, vol. 21, no. 4, pp. 481–504, 2016. [Online]. Available: https://doi.org/10.1007/s00766-015-0229-z

[9] E. Zavala, X. Franch, J. Marco, A. Knauss, and D. E. Damian, "SACRE: supporting contextual requirements' adaptation in modern self-adaptive systems in the presence of uncertainty at runtime," *Expert Syst. Appl.*, vol. 98, pp. 166–188, 2018. [Online]. Available: https://doi.org/10.1016/j.eswa.2018.01.009

[10] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, 2009.

[11] A. Bennaceur and B. Nuseibeh, "The many facets of mediation: A requirements-driven approach for trading-off mediation solutions," in *Managing trade-offs in adaptable software architectures*, I. Mistrík, N. Ali, J. Grundy, R. Kazman, and B. Schmerl, Eds. Elsevier, 2016.

[12] D. M. Yellin and R. E. Strom, "Protocol specifications and component adaptors," *ACM Transactions on Programming Languages and System, TOPLAS*, vol. 19, no. 2, pp. 292–333, 1997.

[13] R. Vaculín, R. Neruda, and K. P. Sycara, "The process mediation framework for semantic web services," *International Journal of Agent-Oriented Software Engineering, IJAOSE*, vol. 3, no. 1, pp. 27–58, 2009.

[14] R. Mateescu, P. Poizat, and G. Salaün, "Adaptation of service protocols using process algebra and on-the-fly reduction techniques," *IEEE Transactions Software Engineering*, vol. 38, no. 4, pp. 755–777, 2012.

[15] L. Cavallaro, P. Sawyer, D. Sykes, N. Bencomo, and V. Issarny, "Satisfying requirements for pervasive service compositions," in *Proc. of the 7th Workshop on Models@run.time*, 2012, pp. 17–22.

[16] P. Inverardi and M. Tivoli, "Automatic synthesis of modular connectors via composition of protocol mediation patterns," in *Proc. of the 35th International Conference on Software Engineering, ICSE*, 2013, pp. 3–12.

[17] N. D'Ippolito, V. A. Braberman, N. Piterman, and S. Uchitel, "Synthesizing nonanomalous event-based controllers for liveness goals," *ACM Trans. Softw. Eng. Methodol.*, vol. 22, no. 1, p. 9, 2013. [Online]. Available: http://doi.acm.org/10.1145/2430536.2430543

[18] A. Bennaceur and V. Issarny, "Automated synthesis of mediators to support component interoperability," *IEEE Trans. Software Eng.*, vol. 41, no. 3, pp. 221–240, 2015. [Online]. Available: http://dx.doi.org/10.1109/TSE.2014.2364844

[19] M. Harman, E. K. Burke, J. A. Clark, and X. Yao, "Dynamic adaptive search based software engineering," in *Proc. of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM*, 2012, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/2372251.2372253

[20] B. Nuseibeh, "On the disappearing boundary between digital, physical, and social spaces: Who, what, where and when?" in *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security, CPSS@AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2, 2017*, 2017, p. 1. [Online]. Available: http://doi.acm.org/10.1145/3055186.3055190

[21] Food and A. O. of the United Nations, "Global food losses and food waste," 2011. [Online]. Available: http://www.fao.org/docrep/014/mb060e/mb060e00.pdf

[22] V. Lim, M. Funk, L. Marcenaro, C. S. Regazzoni, and M. Rauterberg, "Designing for action: An evaluation of social recipes in reducing food waste," *Int. J. Hum.-Comput. Stud.*, vol. 100, pp. 18–32, 2017. [Online]. Available: https://doi.org/10.1016/j.ijhcs.2016.12.005

[23] "IBM Chef Watson," accessed: 09-01-2019. [Online]. Available: http://www.ibmchefwatson.com

[24] P. Angara, M. Jiménez, K. Agarwal, H. Jain, R. Jain, U. Stege, S. Ganti, H. A. Müller, and J. W. Ng, "Foodie fooderson a conversational agent for the smart kitchen," in *Proc. of the 27th Annual International Conference on Computer Science and Software Engineering, CASCON*, 2017, pp. 247–253. [Online]. Available: http://dl.acm.org/citation.cfm?id=3172825

[25] A. Bennaceur, C. McCormick, J. García-Galán, C. Perera, A. Smith, A. Zisman, and B. Nuseibeh, "Feed me, feed me: an exemplar for engineering adaptive software," in *Proc. of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS*, 2016, pp. 89–95. [Online]. Available: http://doi.acm.org/10.1145/2897053.2897071

[26] "Cook's Thesaurus," accessed: 09-01-2019. [Online]. Available: http://www.foodsubs.com

[27] M. Jackson and P. Zave, "Deriving specifications from requirements: An example," in *Proc. of the 17th International Conference on Software Engineering, ICSE*, 1995, pp. 15–24.

[28] F. Rossi, P. Van Beek, and T. Walsh, *Handbook of constraint programming*. Elsevier Science, 2006, vol. 35.

[29] L. Yang, C. Hsieh, H. Yang, J. P. Pollak, N. Dell, S. J. Belongie, C. Cole, and D. Estrin, "Yum-me: A personalized nutrient-based meal recommender system," *ACM Trans. Inf. Syst.*, vol. 36, no. 1, pp. 7:1–7:31, 2017. [Online]. Available: http://doi.acm.org/10.1145/3072614

[30] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowl.-Based Syst.*, vol. 46, pp. 109–132, 2013. [Online]. Available: https://doi.org/10.1016/j.knosys.2013.03.012

[31] R. Ali, C. Solís, I. Omoronyia, M. Salehie, and B. Nuseibeh, "Social adaptation - when software gives users a voice," in *Proc. of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE*, 2012, pp. 75–84.

[32] M. Almaliki, F. Faniyi, R. Bahsoon, K. Phalp, and R. Ali, "Requirements-driven social adaptation: Expert survey," in *Requirements Engineering: Foundation for Software Quality - 20th International Working Conference, REFSQ*, 2014, pp. 72–87. [Online]. Available: https://doi.org/10.1007/978-3-319-05843-6_6

[33] R. M. Karp, "Reducibility among combinatorial problems," in *Proc. of a Symposium on the Complexity of Computer Computations*, 1972, pp. 85–103.

[34] "Epicurious," accessed: 09-01-2019. [Online]. Available: http://www.epicurious.com/recipes-menus

[35] "Better Homes and Garden (BHG)," accessed: 09-01-2019. [Online]. Available: https://www.bhg.com/recipes/how-to/ingredient-substitutions/

[36] A. Bennaceur, T. T. Tun, A. K. Bandara, Y. Yu, and B. Nuseibeh, "Feature-driven Mediator Synthesis: Supporting Collaborative Security in the Internet of Things," *ACM Transactions on Cyber-Physical Systems*, 2017. [Online]. Available: http://oro.open.ac.uk/50803/

[37] R. Edwards and N. Bencomo, "DeSiRE: further understanding nuances of degrees of satisfaction of non-functional requirements trade-off," in *Proc. of the 13th International Conference on Software Engineering*

for Adaptive and Self-Managing Systems, SEAMS@ICSE, 2018, pp. 12–18. [Online]. Available: https://doi.org/10.1145/3194133.3194142

[38] C. Ansótegui, M. Bofill, M. Palahí, J. Suy, and M. Villaret, "Solving weighted csps with meta-constraints by reformulation into satisfiability modulo theories," *Constraints*, vol. 18, no. 2, pp. 236–268, 2013. [Online]. Available: https://doi.org/10.1007/s10601-012-9131-1

[39] A. Schiendorfer, A. Knapp, J. Steghöfer, G. Anders, F. Siefert, and W. Reif, "Partial valuation structures for qualitative soft constraints," in *Proc. of Software, Services, and Systems*, 2015, pp. 115–133. [Online]. Available: https://doi.org/10.1007/978-3-319-15545-6_10

[40] T. H. Tan, M. Chen, J. Sun, Y. Liu, É. André, Y. Xue, and J. S. Dong, "Optimizing selection of competing services with probabilistic hierarchical refinement," in *Proc. of the 38th International Conference on Software Engineering, ICSE*, 2016, pp. 85–95. [Online]. Available: http://doi.acm.org/10.1145/2884781.2884861

[41] A. van Lamsweerde, R. Darimont, and P. Massonet, "Goal-directed elaboration of requirements for a meeting scheduler: problems and lessons learnt," in *Proc. of the 2nd IEEE Intl. Symp. on Requirements Eng., RE*, 1995, pp. 194–203. [Online]. Available: http://dx.doi.org/10.1109/ISRE.1995.512561

[42] P. Sawyer, N. Bencomo, J. Whittle, E. Letier, and A. Finkelstein, "Requirements-aware systems: A research agenda for re for self-adaptive systems," in *Proc. of the 18th IEEE International Requirements Engineering Conference, RE*, 2010, pp. 95–103.

[43] I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli, "Model evolution by run-time parameter adaptation," in *Proc. of the 31st International Conference on Software Engineering, ICSE*, 2009, pp. 111–121. [Online]. Available: https://doi.org/10.1109/ICSE.2009.5070513

[44] S. Fickas and M. S. Feather, "Requirements monitoring in dynamic environments," in *Proc. of the Second IEEE International Symposium on Requirements Engineering, March 27 - 29, 1995, York, England*, 1995, pp. 140–147. [Online]. Available: https://doi.org/10.1109/ISRE.1995.512555

[45] A. Filieri, M. Maggio, K. Angelopoulos, N. D'Ippolito, I. Gerostathopoulos, A. B. Hempel, H. Hoffmann, P. Jamshidi, E. Kalyvianaki, C. Klein, F. Krikava, S. Misailovic, A. V. Papadopoulos, S. Ray, A. M. Sharifloo, S. Shevtsov, M. Ujma, and T. Vogel, "Control strategies for self-adaptive software systems," *TAAS*, vol. 11, no. 4, pp. 24:1–24:31, 2017. [Online]. Available: http://doi.acm.org/10.1145/3024188

[46] M. Salifu, Y. Yu, and B. Nuseibeh, "Specifying monitoring and switching problems in context," in *Proc. of the 15th IEEE International Requirements Engineering Conference, RE*, 2007, pp. 211–220. [Online]. Available: https://doi.org/10.1109/RE.2007.21

[47] J. Kramer and J. Magee, "Self-managed systems: an architectural challenge," in *Proc. of the Future of Software Engineering track, FOSE@ICSE*, 2007, pp. 259–268. [Online]. Available: http://dx.doi.org/10.1109/FOSE.2007.19

[48] I. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos, "The requirements problem for adaptive systems," *ACM Trans. Management Inf. Syst.*, vol. 5, no. 3, p. 17, 2014. [Online]. Available: http://doi.acm.org/10.1145/2629376

[49] E. Letier and W. Heaven, "Requirements modelling by synthesis of deontic input-output automata," in *35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013*, 2013, pp. 592–601. [Online]. Available: http://dl.acm.org/citation.cfm?id=2486866

[50] W. Qian, X. Peng, J. Sun, Y. Yu, B. Nuseibeh, and W. Zhao, "O2O service composition with social collaboration," in *Proc. of the 32nd IEEE/ACM International Conference on Automated Software Engineering, ASE*, 2017, pp. 451–461. [Online]. Available: https://doi.org/10.1109/ASE.2017.8115657

[51] R. Aschoff and A. Zisman, "Proactive adaptation of service composition," in *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-managing Systems*, ser. SEAMS'12, Zurich, Switzerland, 2012.

[52] J. DeMiguel, L. Plaza, and B. Díaz-Agudo, "Colibricook: A CBR system for ontology-based recipe retrieval and adaptation," in *Proc. of the 9th European Conference on Case-Based Reasoning*, 2008, pp. 199–208.