# Recipe Enrichment: Knowledge Required for a Cooking Assistant

Nils Neumann and Sven Wachsmuth
*Research Institute for Cognition and Robotics (CoR-Lab), Bielefeld University, Universitätsstr. 25, 33615 Bielefeld, Germany*

Keywords: Knowledge Representation and Reasoning, Knowledge-based Systems, Cognitive Assistance, Assisted Living, Applications of AI.

Abstract: The preparation of a meal consisting of multiple components with different timing and critical synchronization points is a complex task. An automated system assisting a human in the preparation process needs to track the progress state, prompt the next recipe steps, control kitchen devices, monitor the final preparation time, and deal with process deviations. This requires a detailed process representation including knowledge about states with critical timing, control signals of devices, preparation steps and cooking times of ingredients, and necessary user attention. At the same time, the system should be flexible enough to allow the free combination of component recipes and kitchen devices to support the preparation of complete menus independent of a specific kitchen setup. To meet these requirements, we propose a method to automatically enrich simple component recipes with process-specific information. The resulting detailed process description can be processed by standard scheduling algorithms to sequence the preparation steps of complex meals. The control information for kitchen devices is already included in the process description, so that monitoring of the progress becomes possible. The reasoning process is driven by so-called *action templates* that allow to decouple knowledge on recipes, ingredients, and kitchen devices in seperate re-usable knowledge bases.

## 1 INTRODUCTION

Preparing a meal consisting of multiple components is a task with a high cognitive workload, especially for inexperienced human cooks. It requires planning, coordination, multitasking as well as memorizing and accessing knowledge about ingredients, cooking devices and associated actions. We aim at the development and implementation of an automated intelligent assistance that supports humans in the cooking process at their regular kitchen environment. In this paper, we address the challenge of formalizing and enriching component recipes so that process descriptions for preparing complete meals can be automatically generated. These can be utilized for monitoring, control, and feedback loops during the human-centered cooking process in order to significantly reduce his or her cognitive workload. In order to close the representational gap between regular fixed, predefined recipes for complete meals towards a generic process description of a human cooking process, we need to model implicit knowledge that an experienced cook would add by his or her "skilled practice, the senses and memory" (Sutton, 2018). As a key concept, we propose an action-centered representation consisting of a composable template structure.

In a regular predefined recipe, implicit knowledge is already considered during its creation, such as causal relationships between the sub-tasks, use of the devices, and the attention resource required by the cook. However, meals are typically composed of multiple components, with one sub-recipe for each component, that partially interfere with each other. Thus, recipes cannot be freely combined without solving conflicts between device usage and task order. In order to find a generic approach that allows free combination of component recipes, more information for each sub-task are required, like its necessary attention or logical connection between tasks. Having a composable representation in place would allow to scale, individualize, and adapt the cooking process to the needs, preferences, and expertise of the user as well as to the availability of the kitchen equipment.

In the cooking domain, several assistant systems (An et al., 2017; Sato et al., 2014) exist following different approaches. Some cooking assistants like the "KogniChef" (Neumann et al., 2017) model the skills together with dependencies between the tasks and additional information for device coordination inside the recipe. This provides a good solution for cooking

Neumann, N. and Wachsmuth, S.
Recipe Enrichment: Knowledge Required for a Cooking Assistant.
DOI: 10.5220/0010250908220829
In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART 2021) - Volume 2*, pages 822-829
ISBN: 978-989-758-484-8

fixed sequential recipes but does not allow the cook to prepare component variations or different meals in parallel. Another cooking assistant "Cooking navi" (Hamada et al., 2005) is utilizing action units that have dependencies between each other and uses devices and the user as resources to plan the execution order of the action units. A plan is provided over a touch panel as the user interface, without taking the feedback of kitchen devices into account.

The Assistive Kitchen (Beetz et al., 2008) and RoboEarth (Tenorth et al., 2012), among others, prepare a meal with a robot. Therefore, they use a recipe that provides a set of steps, add missing information and additional micro actions which contain motion primitives that can be executed by the robot. These systems provide an approach for task representation using an ontology and multiple small steps but focus on a different goal with other required information, due to coordinating a robot and not a human.

The TAAABLE system, from (Badra et al., 2008), focuses on providing cooking recipes with a case based system on a semantic wiki. It receives requests with constrains for the recipe like desired ingredients and provides a text based solution for the recipe. It also forms the basis for the Computer Cooking Contest (CCC) (Nauer and Wilson, 2015). Teams compete to find the best solution for given challenges. 2015 challenges like providing a sandwich recipe or making a cocktail with specified wanted and unwanted ingredients were solved. The EVER project (Bergmann et al., 2017) creates semantic workflows based on textual sources, using similarity-based retrieval methods to find the best matching workflow. They also participated multiple times in the CCC. While the TAAABLE system, EVER project and the CCC generate or adapt recipes for humans, we focus on enriching recipes for the execution with a cooking assistance, that requires additional information about the recipe steps.

In this paper, a recipe representation utilizing *action templates* is described that closes the gap between a regular predefined recipe and a composable recipe for a cooking assistant supporting a human cook. The action templates introduce a new abstraction layer, which captures device-dependent interaction patterns and feedback loops between a human user and the partially automated device control (comparable to the robot's micro actions in the Assistive Kitchen). They further allow a free combination of different component recipes considering the required implicit knowledge, device setup and user skills, which enables a high individualization and flexibility for the cook.

## 2 PROBLEM STATEMENT AND SYSTEM DESCRIPTION

Regular predefined recipes already encode fixed implicit knowledge about tasks, task dependencies, ingredients, actions, and kitchen devices. Either they leave their combination with other recipes to the skills of the cook or need to be completely rewritten for any changing component or device. They do not consider recipe variations or interference with other recipes. Consequently, the sequence of tasks is defined from the perspective of the creator of the specific recipe and his or her assumptions about a standard cook. Enabling an automated cooking assistant to support an individual human cook, while preparing multiple component recipes together as a meal requires a higher flexibility.

Therefore we propose an action-centered approach that explicitly represents the knowledge implicitly coded before. Figure 1 shows the difference between a simple component recipe (provided as an input to the system) and a process description (provided as an output of the system). The method proposed in this paper is able to automatically enrich such a recipe utilizing generic knowledge bases.

The action-centered approach is embedded in a complete system for assisting users in a cooking process at home. The main components of this system are shortly described in the following. In this paper, we focus on the recipe enrichment method providing the process description:

- **Knowledge Representation:** provides the required recipe information from three sources (action-template ontology, ingredient representation, recipe representation) and combines these into multiple component recipes with different types of preparation, which are usable at the same time by the cooking assistance.
- **Scheduler:** schedules multiple component recipes together providing an executable sequence of tasks, based on the available devices and the different types of preparation.
- **Device Control:** provides the available devices for the planning component, monitors and executes tasks for the devices.
- **Monitoring:** supervises and controls the execution of the recipe, handles input from the user interface and device control, detects deviations from the plan and starts re-planning if necessary.
- **User Interface:** provides information and interacts with the user.

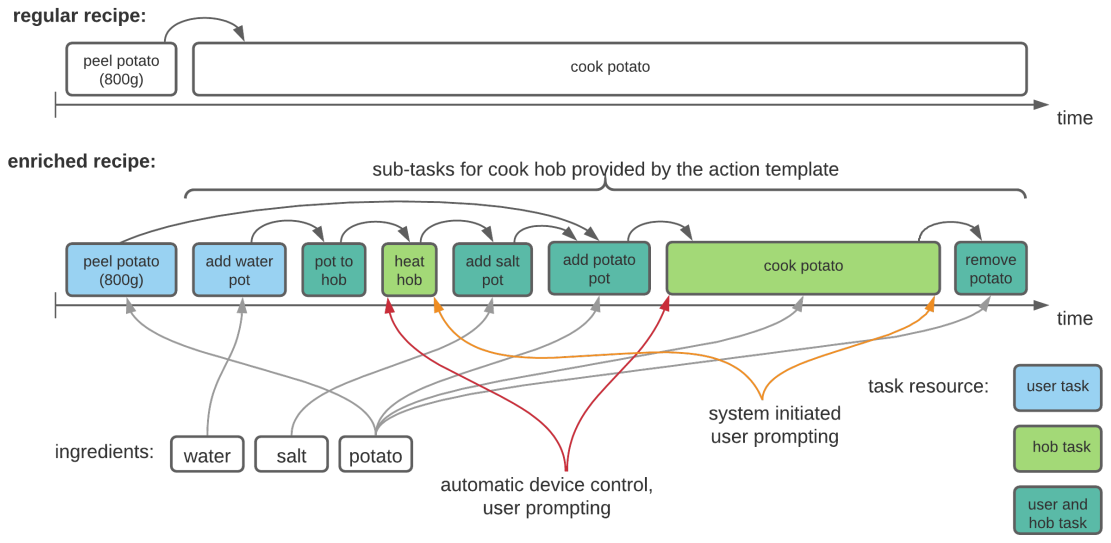

Figure 1: The upper part of the picture shows a regular recipe for the preparation of cooked potatoes. The lower part of the picture shows the enriched recipe for cooked potato with the action-centered approach that enables the use by the kitchen assistant. Here the "cook potato" task is split into several smaller tasks. Each task includes the necessary resource information for the execution (task color), the logical dependency between the task (arrows over the tasks), device control commands for automatic device control and prompting information for the user when the state of the device changes.

## 2.1 Modeling Composable Recipes for Assisted Cooking

Most regular (human-defined) recipes are designed by combining an average of ten actions together with an average of nine ingredients (Yamakata et al., 2017; Salvador et al., 2017). Only a limited number of around 100 common action-verbs are used in a cooking scenario, as shown in the EPIC-KITCHENS data set (Damen et al., 2018). Therefore, we keep our action-centered approach with a similar subdivision of action types.

Our action-centered approach consists of three different knowledge sources which are, then, combined to create an executable component recipe by utilizing the type of action as binding concept connecting them:

- **Action-template Ontology:** an action template is a concept in the action-template ontology defining the action types to realize the appropriate tasks given by the recipe representation which is enriched by the ingredients from the ingredient ontology. Therefore the action templates can include multiple action-options to realize a task in a recipe, e.g. cook with steamer or hob, each with previous implicit sub-actions and their dependencies, necessary devices/utensils required for each action-option. It further defines how the sub-actions are parameterized applying the information from the ingredient representation and recipes, as shown in Figure 1 for *cook hob* with the parameterization of the potato ingredient. The action template instantiates the task from the recipe with (multiple-) actions executable by the cooking assistant based on the devices, cook attention, devices commands, and device interaction types.
- **Ingredient Representation:** an ingredient inside the ingredient representation is a concept that defines all executable action templates for the specific ingredient and their parameterization (duration etc.).
- **Recipe Representation:** a recipe in the recipe representation contains the specific tasks which must be carried out to prepare and cook the defined component recipe. Each task from a recipe consists of a reference to an action template, a list of ingredients with quantities and the logical dependencies to other tasks. If no further parameters are set, the ingredient information for the specified action template is used to parameterize the task.

Combining the component recipes selected by the cook, as described in the recipe representation, creates the menu. Each step from a component recipe is represented as a task with its logical dependencies. These dependencies are described as connections to its predecessor and successor tasks, instead of a chronological sequence as frequently found in regular recipes. This creates a higher flexibility in combining component recipes, but means that the enriched process description – which combines all component recipes – must explicitly include all information required for a later scheduling of the tasks. As a consequence, each component recipe can be composed individually making it easier to extend the knowledge base.

**Deriving Ingredient Information for Recipes.** Inside the ingredient representation all feasible types of action are defined for each ingredient and necessary parameters are parameterized with default values independent of the recipe. These action types are linked to the action templates and provide all information for the execution by a cooking assistant, like the abstract device interaction type, the necessary subactions for the execution or scaling function for different ingredient quantities. For an instantiation in the process description, each action-ingredient pair of a task from the recipe is bound to an action type in the ingredient representation and the task is enriched with missing information from the ingredient representation and the associated action template. Through this distributed representation of knowledge, the redundancy is reduced, while keeping the opportunity to adjust parameter specifically for a recipe. As a consequence, only very few information are required in a component recipe, such that its creation in a machine-readable format is a nearly effortless task. In this way, the tasks in the component recipes just need recipe specific information (e.g. divergent cooking times, quantities of ingredients). All general knowledge is taken from their default parameters. As an example, duration times (relative to the quantity) are defined in the action templates (by using scaling functions). This allows an automated scaling of different quantities in the recipe.

**Deriving Sub-tasks for Monitoring, Prompting and Device Control.** Enabling a cooking assistant to support a human cook during the handling of kitchen devices requires a more detailed representation than typically encoded in recipes, as shown in Figure 1. These information are coded in the action templates and describe the execution of a task in a more precise way. The action templates are parameterized with the information from the recipe utilizing the task representation as an action-ingredient pair. Combing the recipe and ingredient information (duration, temperature, dependency between tasks, ingredients and quantities) with the action templates that provide information like user attention, device parameters, and additionally required actions, creates the detailed actions that are used by the cooking assistant for prompting and monitoring the cooking progress.

Splitting a task into multiple smaller actions provides advantages in terms of utilizing more precise information of the cooking process and a more exact planning. Considering the limited user attention, e.g. cooking in a steamer has a changing attention level during the execution. Adding and removing has a high user attention, while cooking has no user attention. Having smaller actions also creates device control and feedback loops based on the combination of device feedback and abstract device interaction types as coded in action templates. These abstract semantic information (e.g. *Add, Remove, Execute, Interact*) together with other action parameters (e.g. duration, temperature, device mode) can be executed and monitored by the device control component, which receives the device's feedback over a network interface (e.g. state of the device (program, door contact switch)). Therefore, the *device interaction type* from the action template, combined with the device feedback, as instantiated in the process description is utilized to automatically monitor the task progress, e.g. while an action with an *Add* interaction type is running, opening and closing an oven, can be interpreted as putting something inside. Using a limited set of abstract interaction types reduces the number of cases that can occur. This enables an automatic device control and progress detection, reducing the confirmation actions for the user and creating a more appropriate feedback for the human cook.

**Deriving Planning Options for Task Scheduling.** Preparing multiple component recipes as a meal requires solving resource conflicts. While the necessary information are available in the process description, there is often more than one option to perform an action with a specific ingredient considering the available devices, (e.g. boil noodle in a hob or steamer). Therefore the action templates have a hierarchical structure, that contains all action-options (e.g. different devices/utensils) for the specific action type (e.g. action template *cook* has action-options *cook hob* and *cook steamer*). These possible options of preparation can be limited in the ingredient or recipe representation by the specification and parameterization of possible devices/utensils. In this way, the scheduling algorithm is able to find the best plan depending on the set of chosen component recipes, enriched by the different knowledge sources, the possible action-options and the available devices provided by the device control component.

The separation of tasks and action templates further enables a personalization of the user actions, storing previous cooking performances in relation to the action and ingredient. Also, their personal preferences, like preferred devices and recipe combinations, can be considered in future planning processes.

Through the enrichment of the recipe, we know the logical dependencies between tasks and actions, the necessary kitchen devices and user attention for each action. This defines a valid process description for computing the execution order of the actions in the component recipes selected. Considering the cook and the kitchen devices as resources, with the user attention level as workload for the resources and the individual recipes as projects, we can interpret it as multi-mode resource-constrained multi project scheduling problem. We solved the scheduling problem with the optaplanner tool (De Smet et al., 2016), while considering rules that take the dependency between actions, the workload of the resources (device and user), the maximum duration of the cooking process and the synchronization of the end time for the individual recipes into account.

## 2.2 Generating Process Descriptions from Composable Recipes

In the following, we explain how the process description (e.g. prepare potatoes with meatballs and cauliflower) is computed in the knowledge representation from scalable and composable component recipes, which is then passed to the scheduler. The cooking assistant is implemented as a distributed system, as described in the section 2, offering interfaces via the MQTT network protocol using the interface description language protocol buffers to serialize the data.

At first, the cook selects the composable component recipes to be prepared together with the number of servings. For each action in a selected component recipe, the content information required for its coordination in the cooking process are automatically deduced from the ingredients in the ingredient representation, if they are not specified in the recipe.

A recipe, as shown in Figure 2, has general information (title, number of servings) and a list of linked tasks. The number of servings allows to scale a recipe consistently relative to the number of servings chosen from the cook. The list of tasks includes unique identifiers for all actions required for a recipe. The action key in each task references to an action in the appropriate ingredient representation, if an ingredient is used in the step and the ingredient can be used with the corresponding action template. Furthermore, each task contains the ingredients, their quantity, and a list of predecessor tasks using the unique identifier from the recipe task list. Optional parameters for each task

```
recipeName: Cooked Potatoes
servings: 4
lastTask:
  - cookpotato
tasks:
  cookpotato:
    action: cook
    ingredients:
      potato:
    preTask:
      - peelpotato
  peelpotato:
    action: peel
    ingredients:
      potato: 800 g
```

Figure 2: Recipe “Cooked Potatoes”, with two tasks “cookpotato” and “peelpotato” that have a dependency, where “peelpotato” must be finished before “cookpotato” can start. The ingredient potato with 800g is used for 4 servings. The actions “cook” and “peel” from the recipe tasks are used as references for the actions in the ingredient.

can be a specific duration if, e.g., the cooking time is shorter than normal for an ingredient, images and videos if they are specific for the recipe. They can also contain, required and preferred devices, utensils and additional action parameters if they are required by the action template.

```
ingredientname: potato
name: potato
unit: g
actions:
  peel:
    - duration: 2 min/100g
      description:
        en: "peel the potato ..."
      utensils:
       - knife
  cook:
    - device: steamer
      duration: 17 min
    - device: hob
      duration: 25 min
```

Figure 3: Ingredient “potato” with parameters for actions that can be performed with the ingredient. The task “peelpotato” from the recipe in Figure 2, with action “peel” and ingredient “potato” use the parameter from the “peel” action in the ingredient “potato” for the parameterization of the peel action template. The duration for the action “peel” is relative to the quantity of potatoes in the recipe. The action “cook” has parameters for the preparation options with a steamer (17 min) or with a hob (25 min).

Missing non-recipe-dependent parameters are added from the ingredient representation, as shown in Figure 3. This template contains basic information about the ingredient, as well as parameter for each processable action type. E.g. ”peelpotato” from the recipe task in Figure 2 uses 800g potatoes for 4 servings. This information is combined with a duration of 2 min/100g from the ”peel”-action in the ingredient representation and a scaling function from the action template to calculate the duration with the used amount of the ingredient. The scaling function from the action template is necessary, since the scaling can differ depending on the action (e.g. constant value for cooking, linear scaling for peeling).

Subsequently, the list of tasks, completed by the ingredient representation, is expanded by applying the action templates to the named appropriate tasks. The action templates enrich the recipe and create the actions for the cooking assistant as shown in Figure 4. Each action template may contain several actions as pre-/post-actions, that split the recipe task in multiple more fine graded actions. They further define if pre-/post-actions are parameterized with fixed values or if their parameters are passed through from the recipe.

Additionally, some action templates model several options to carry out a task. E.g. the action template “cook” can be performed using either a steamer or a hob as shown in Figure 4. Both are modeled as sub-action templates derived from the “cook” template. All valid options that fulfill the constraints of available devices and recipe specifications are taken into account for the subsequent scheduling task.

After connecting the tasks from the recipe with the action-templates, the relation between the tasks implies dependencies between first and last (pre- or post-) actions of the corresponding action templates as shown in Figure 4, part 3. The user attention (depicted for each action by the stick figure) is provided by the action-templates and allows a parallel execution of actions as long as their user attention together do not exceed the maximum workload for the resource. The action-templates further specify how the duration of the action scales with ingredient quantities defined in the recipe. A post action priority ensures the immediate execution of time-critical tasks following the action. Additionally, categorized types of (user-) device interactions, like *add*, *execute*, *remove*, *interact*, and *preheat*, allows an automated progress detection based on sensor data of the devices (e.g. contact switches, temperature drops). This can be utillized for generating feedback loops for the user in the device control component. For prompting and control, they include information like device commands, possible cutting states, temperature, or utensils required to expand the actions.

Utilizing the separated knowledge sources, individual user profiles can also be maintained. If registered users have used the cooking assistant before, their preferences for multiple action-options, their preferred recipe combinations, and their preparation times required are saved in relation to the action and ingredient. Based on this information, recipe combinations can be recommended, cooking instructions can be adjusted to user skills and preferences, and preparation times for an action can be estimated more precisely.

After enriching the recipe with necessary information for the execution by a cooking assistant, the recipe is planned by the scheduling component, as shown in Figure 5. The scheduling process takes the available resources (devices provided by the device control component and cook provided by the user interface) and dependencies between tasks into account and generates an executable sequence of tasks with the available devices. The planned sequence of tasks is executed and supervised by the monitoring component. Therefore, it uses feedback from the user interface and the device control component. If the execution of the recipe deviates too much from the planned recipe, a re-planning is started. If possible, the confirmation of the recipe task is automatically accomplished by the device control, else the confirmation is carried out by the cook via the device interface. For the automatic task confirmation the device control component uses the sensor input from the devices, e.g. door contact switch, selected program, temperature, in combination with the device interaction types. Using the device interaction types abstracts the action independently of the devices, e.g. hob, oven, steamer, and defines a fixed amount of interaction possibilities. The detection of the device interaction types may differ for different devices, e.g. ”Add something in the steamer” can be detected by the door contact switch, while ”Add something in a pot on the hob” requires other device information due to the missing door contact switch. By knowing which interaction type has to be carried out, sensor information can be interpreted in the appropriate context.

Due to the action-options with different devices, a more flexible planning for multiple component recipes is possible. E.g. if the steamer is used for the cauliflower, the potato can alternatively be prepared with the hob. By the division of tasks into smaller sub-tasks, as shown in Figure 5 for the cook potato task, a more accurate time management for the cook is made possible, which can enable better planning options.

## 3 VALIDATION OF THE RECIPE ENRICHMENT

The approach was validated in a prototypical kitchen environment with the cooking assistant as described in the section 2. The setup included real kitchen devices which were automatically registered as resources in the system. The knowledge base consisted of a set of 3 carbohydrate side dishes (C), 3 vegetable side dishes (V) and 3 main components (M), with multiple preparation possibilities, which were

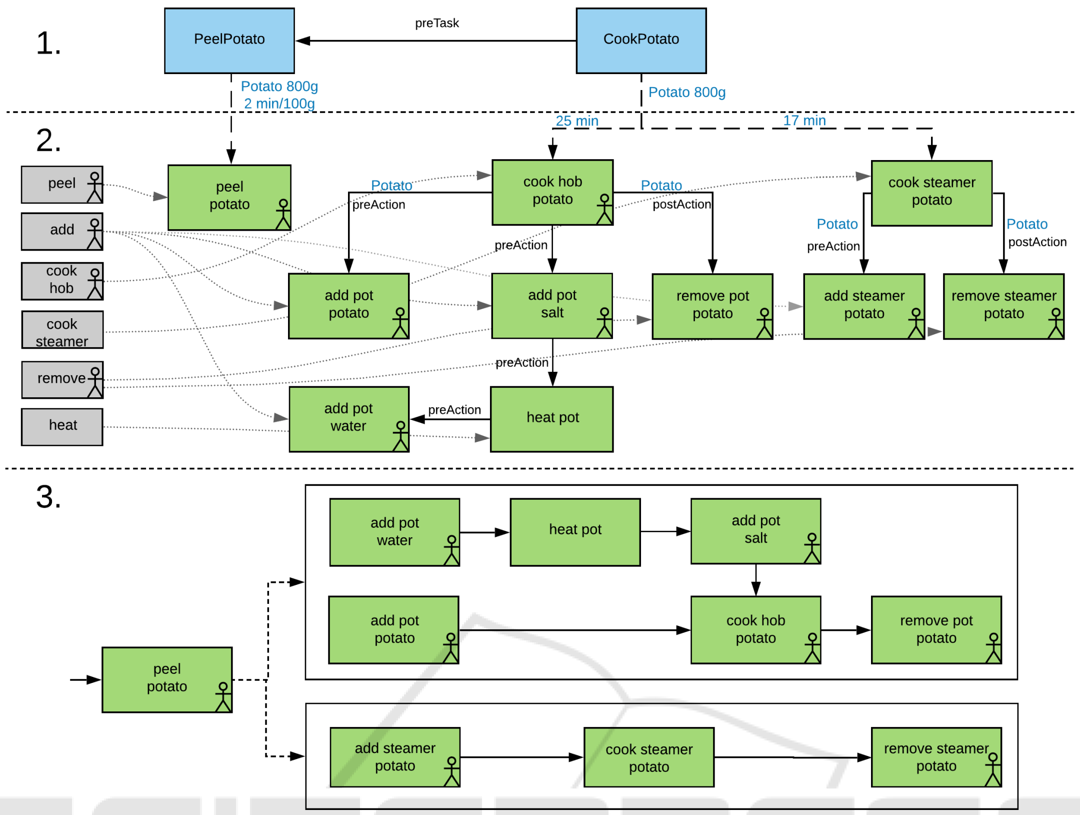

Figure 4: The recipe tasks enriched with the ingredient parameter (1. blue), are combined with the corresponding action templates (2. grey), creating the actions (2. green) with their pre-/post-actions. The actions are parameterized with the recipe parameter (blue text) and the action template information (gray arrow, e.g. stick figure for user attention). The first row of each action shows the action type and used device/utensil. The second line the ingredient. While “peel potato” only has one option and no pre-/post-actions, “cook potato” has two options with pre-/post-actions. 3. The dependencies from the recipe are transferred onto the actions and the “peel potato” action on the left now points to both possible action-options, where the scheduler have to choose one for the cooking process.

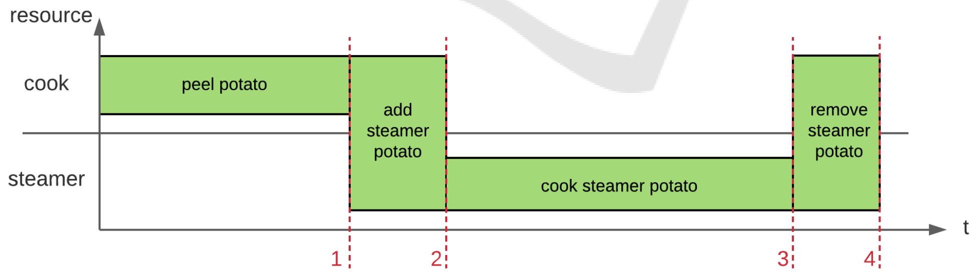

Figure 5: Planning example of one component recipe. Starting from the generated plan in Figure 4 part 3. The recipe is planned depending on the available devices. In this example the kitchen only contains a steamer and no hob. Therefore the possibility with the hob (Figure 4 part 3 upper option) can not be executed with the available setup and is removed. The remaining tasks are planned, as shown, due to the utilization of resources (steamer and cook), and the dependency between the tasks. During the execution of the recipe, the completion of the first cooking step ”peel potato” (red 1) must be confirmed by the cook, since no devices are used. The steps ”add-/remove- steamer potato” (red 2/4) requires both the cook and the steamer as resources, in this cases the task execution is detected by the device due to the interaction with the door and the next step is started. The cooking step ”cook steamer potato” only needs the steamer as resource and requires no attention from the cook. In this time the cook could perform other tasks from other component recipes if multiple component recipes are planned together. Due to the feedback from the device the end of the task (red 3) is detected automatically.

provided by a professional cook. These component recipes were freely combined to valid recipes, with one component from each set, considering the available devices (changing combinations of hob, steamer, oven). The average number of tasks in a menu was 33.6 (C: 14.3, V: 4.6, M: 14.6). After the enrichment of the recipes into a complete description of the cooking process, the average task number changed to 44.3 (in case of multiple options the average number of tasks was counted). Multiple freely combined recipes were prepared as system validation, both with and without food, resulting in a functional cooking assistance, in terms of planning, re-planning in case of deviations, device control and automatic confirmation of user interactions from the devices.

## 4 CONCLUSION

While regular (human readable) recipes look well-structured and simple, the cooking process itself frequently includes many pitfalls and unwritten (but necessary) small pre- and post-actions which determine the success of cooking. Therefore, today's available cooking assistants are designed either for a very specific device or for predefined hand-modeled recipes which are completely decoupled from kitchen devices. In comparison to other cooking assistants our action templates enriches the recipe steps not only with necessary information for visualization, planning, micro actions, and commands for device coordination, but also with information about user-device interaction and optional preparation alternatives. In this paper, we propose action templates as a backbone for modeling component recipes that can be enriched and combined to full-blown descriptions of the human-device interactions during a cooking process and show its potential and benefits.

## REFERENCES

An, Y., Cao, Y., Chen, J., Ngo, C.-W., Jia, J., Luan, H., and Chua, T.-S. (2017). Pic2dish: A customized cooking assistant system. In *Proceedings of the 25th ACM International Conference on Multimedia*, MM '17, page 1269–1273, New York, NY, USA. Association for Computing Machinery.

Badra, F., Bendaoud, R., Bentebibel, R., Champin, P.-A., Cojan, J., Cordier, A., Despres, S., Jean-Daubias, S., Lieber, J., Meilender, T., Mille, A., Nauer, E., Napoli, A., and Toussaint, Y. (2008). Taaable: Text mining, ontology engineering, and hierarchical classification for textual case-based cooking.

Beetz, M., Stulp, F., Radig, B., Bandouch, J., Blodow, N., Dolha, M., Fedrizzi, A., Jain, D., Klank, U., Kresse, I., Maldonado, A., Marton, Z., Mosenlechner, L., Ruiz, F., Rusu, R., and Tenorth, M. (2008). The assistive kitchen — a demonstration scenario for cognitive technical systems. pages 1 – 8.

Bergmann, R., Minor, M., Müller, G., and Schumacher, P. (2017). Project ever: Extraction and processing of procedural experience knowledge in workflows. In *ICCBR (Workshops)*, pages 137–146.

Damen, D., Doughty, H., Farinella, G. M., Fidler, S., Furnari, A., Kazakos, E., Moltisanti, D., Munro, J., Perrett, T., Price, W., and Wray, M. (2018). Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*.

De Smet, G. et al. (2016). Optaplanner user guide. *Red Hat and the community. URL http://www. optaplanner. org. OptaPlanner is an open source constraint satisfaction solver in Java*.

Hamada, R., Okabe, J., Ide, I., Satoh, S., Sakai, S., and Tanaka, H. (2005). Cooking navi: assistant for daily cooking in kitchen. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 371–374.

Nauer, E. and Wilson, D. C. (2015). Computer cooking contest.

Neumann, A., Elbrechter, C., Pfeiffer-Leßmann, N., Kõiva, R., Carlmeyer, B., Rüther, S., Schade, M., Ückermann, A., Wachsmuth, S., and Ritter, H. J. (2017). "kognichef": A cognitive cooking assistant. *KI - Künstliche Intelligenz*, 31(3):273–281.

Salvador, A., Hynes, N., Aytar, Y., Marín, J., Ofli, F., Weber, I., and Torralba, A. (2017). Learning cross-modal embeddings for cooking recipes and food images. pages 3068–3076.

Sato, A., Watanabe, K., and Rekimoto, J. (2014). Mimicook: A cooking assistant system with situated guidance. pages 121–124.

Sutton, D. (2018). *Cooking Skills, the Senses, and Memory: The Fate of Practical Knowledge**, pages 88–109.

Tenorth, M., Perzylo, A. C., Lafrenz, R., and Beetz, M. (2012). The RoboEarth language: Representing and Exchanging Knowledge about Actions, Objects, and Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, MN, USA. Best Cognitive Robotics Paper Award.

Yamakata, Y., Maeta, H., Kadowaki, T., Sasada, T., Imahori, S., and Mori, S. (2017). Cooking recipe search by pairs of ingredient and action —word sequence v.s. flow-graph representation—. *Transactions of The Japanese Society for Artificial Intelligence*, 32:71–79.