



### Summary

su-CHEF is an application of context-aware dynamic service composition. The system employs a classical AI planner to facilitate accurate and easy cooking without recipe books, and allow for transparent adaptation of recipes to changing cooking conditions.

**Key features:**

- su-CHEF builds recipes at run-time based on the environment where the dish is to be cooked, taking into account the characteristics of household devices, cooking utensils, and ingredients available.
- It uses **greplets**: goal-driven recipes, describing not the actions to be taken in a cooking process, but the desired state of items to be reached in each of the intermediate stages of the process.
- su-CHEF deals with **unexpected circumstances**, such as device failures or unanticipated recipes can be dynamically recomposed when environmental parameters change.

### 1. Environment Description Builder

The builder collects information about the available ingredients, devices, utensils and their states, user preferences, and actions possible in the kitchen domain, and represents this using STRIPS syntax.

```

;; Available devices:
(device oven)
(oven oven_1)
(available oven_1)
(operational oven_1)
(device_temperature oven_1 0)
(can_bake oven)
(can_warm_up oven)
    
```

Actions (e.g. "chop") are state transition functions, defined by a precondition list, an add list reflecting the new state of an item after the operation has been carried out on it (e.g. "onion is now chopped"), and a delete list representing the state that it no longer is in (e.g. "onion is no longer in one piece/whole")

```

;; Sample operator
(def-strips-operator (chop ?u ?i ?x ?y ?z)
  (pre
    (ingredient ?i) (width ?x) (height ?y) (length ?z)
    (whole ?i) (can_chop ?u))
  (add (chopped ?i ?x ?y ?z))
  (del (whole ?i)))
    
```

### 2. Greplet Manager

Greplets, describing the state to which ingredients should be brought by the end of each cooking step, are also represented declaratively in STRIPS language. An example greplet for the first state of lasagne cooking, which is that of ingredient preparation.

```

(define (preparation_step)
  (chopped onion)
  (sliced mushroom slice_thickness)
  (grated mozzarella grate_thickness)
  (whole tomato))
    
```

The greplet manager discovers greplets, given the environment description from:

- greplet cache --- maintained by the system to reduce unnecessary remote lookups. It stores a list of recently used greplets and operates an LFU replacement policy.
- greplet database --- stores a larger number of greplets, but is more costly to look up - in terms of time, as it is remote, and money, as it may charge for queries.
- other sources, such as a p2p network, a CDN, or on-package greplet RFID tags

### 3. Recipe Builder

Dynamic recipe composition is mapped to the classical planning problem - the problem of finding a finite sequence of actions that will transform a given initial state - available ingredients - to a state that satisfies a given goal - wanted dish

```

graph LR
    subgraph Domain
        direction TB
        D["domain description  
(ingredients, devices, actions)"]
        G["goal  
(greplet)"]
    end
    D --> AI[AI Planner]
    G --> AI
    AI --> P["plan  
(recipe)"]
    
```

Prototype features:

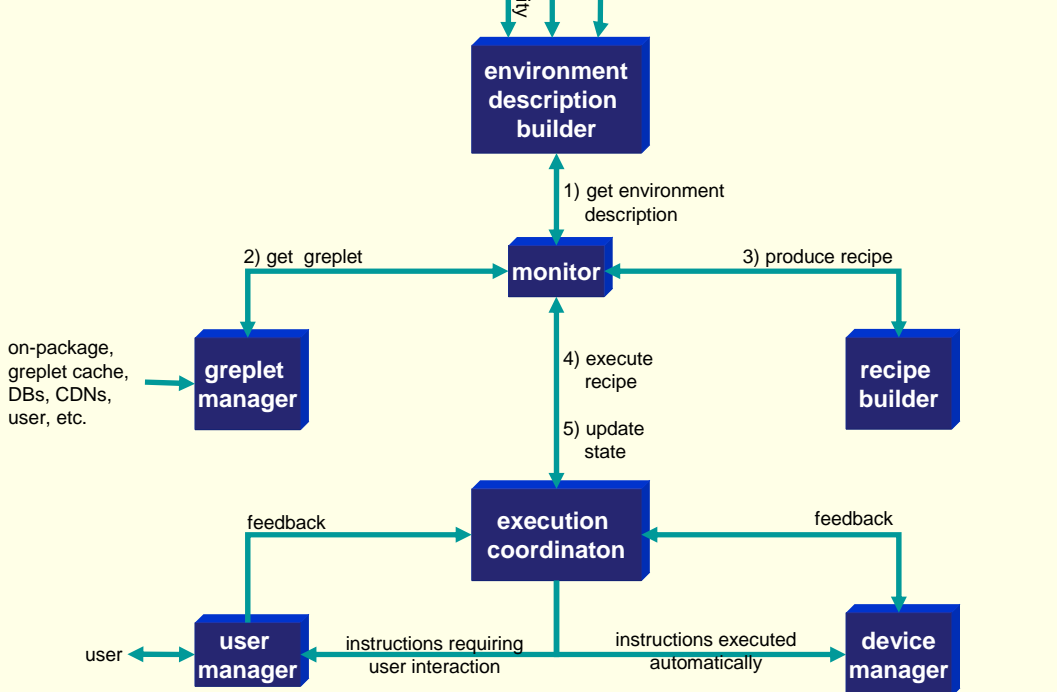
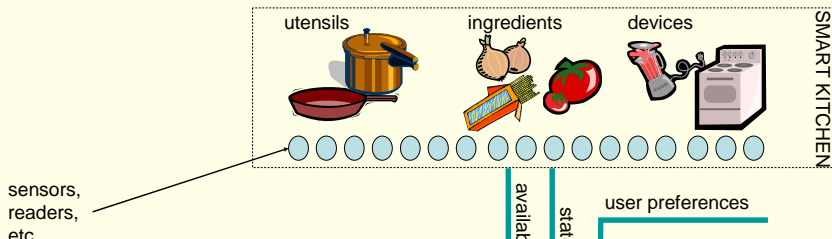
- uses the TLPlan planning system - supporting all the complex control structures involving concurrency, iteration and nondeterministic choice to generate expressive compositions. It allows for concurrently executing actions with varying durations.
- su-CHEF produces all its internal representations independently of the recipe builder's own language, and then it is the responsibility of translator components, specific to the recipe builder used.

### Research Challenges

- Representation of the cooking environment
- Adaptation of the cooking process to different or changing environments
- Adaptation of recipe execution based on feedback from devices and sensors
- Coordination of device and user-based tasks
- Ease of use
- Fault tolerance
- Self management of the cooking process

### Sample Recipe produced by su-CHEF in human readable form

- Place onions in chopper
- Place mushrooms in slicer
- Place mozzarella in grater
- Put one tablespoon of oil in frying pan
- Place frying pan on hot plate
- Chopper: Chop onions in 50x50x50mm cubes
- Slicer: Slice mushrooms in 60mm slices
- Grater: Grate mozzarella at 0.5mm thickness
- Hot plate: Turn on at maximum temperature
- Hot plate: Wait for 2 minutes 45 seconds
- Add onions in pan
- Hot plate: Wait for 3 minutes 30 seconds
- Add minced beef in pan
- ...
- Lay lasagne sheet in baking tray
- Place 1/4 of pan contents in baking tray
- Lay lasagne sheet in baking tray
- Top with mozzarella
- Oven: Turn on at 200 degrees
- Oven: Wait for 8 minutes
- Place baking tray in oven
- Oven: Wait for 47 minutes
- Take baking tray out of oven



### 4. Execution Coordinator

- Translates the STRIPS-based cooking instructions produced by the recipe builder to BPWL4WS plan, deployed on BPWS4J, describing the cooking process.
- Assigns tasks to either the User or the Device Manager - which are proxies for user-executed and device-executed tasks respectively.

```

<process name="LasagneCookingProcess">
  ...
  <sequence name="main_cooking_sequence">
    <receive name="receive" partner="caller">
      portType="tns:ExecutionCoordinatorPT"
      operation="invokeExecution"
      variables="recipe" createInstance="yes"/>
      ...
      <flow><sequence name="preparation_step">
        ...
        <invoke name="invoke" partner="DeviceManager">
          portType="dn:DeviceManagerPT"
          operation="invokeDeviceManagerProxy"
          inputVariables="chop_onion_request"
          outputVariable="chop_onion_response"/>
          ...
        </sequence></flow></sequence> </process>
    
```

### 5. Monitor Module

Monitor module:

- Coordinates and observes the cooking process
- Detects and evaluates changes in the environment

Based on the changes in the environment it triggers two types of adaptation:

- Adaptation of the execution of a recipe
  - e.g. when the state of an ingredient changes to indicate that a step is complete such as "onion sauteed" - this update is to the execution coordinator.
- Adaptation of the recipe itself
  - e.g. when an ingredient or device is no longer available, such as "minced beef disappeared" - the monitor triggers re-planning to compose a new recipe that can be carried out in the current environment
  - requires a search for a new, suitable greplet to be launched. Along with the new environment, the new greplet is passed on to the recipe builder for a recipe to be produced and executed as normal.

### References

- M. Vukovic and P. Robinson. Adaptive, Planning Based, Web Service Composition for Context Awareness. In: Advances in Pervasive Computing, Volume 176, (2004) 247-252.
- F. Bacchus and F. Kabanza. Using Temporal Logic to Control Search in a Forward Chaining Planner. In Proceedings of Second International Workshop on Temporal Representation and Reasoning, Melbourne Beach, Florida, 1995.
- Durum wheat semolina and alimentary pasta - Estimation of cooking quality of spaghetti by sensory analysis. Technical Report ISO 7304:1985, ISO, 1985.
- M. Langheinrich, F. Mattern, K. R. omer, and H. Vogt. First Steps Towards an Event-Based Infrastructure for Smart Things. Ubiquitous Computing Workshop (PACT 2000), Oct. 2000.
- S. Ramasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In Proc. of the 2001 Conf. on apps, tech., arch., and protocols for computer communications, pages 161-172. ACM Press, 2001.
- U. Saif, H. Pham, J. M. Paloska, J. Waterman, C. Terman, and S. Ward. A case for goal-oriented programming semantics. In Fifth Annual Conf. on Ubiquitous Computing, Workshop on System Support for Ubiquitous Computing, 2003.
- D. Wu, E. Strin, J. Hender, D. Nati, and B. Parsia. Automatic Web Services Composition Using SHOP2. In 13th Int. Conf. on Automated Planning & Scheduling, Workshop on Planning for Web Services, Italy, June 2003.