

A Cooking System in ROPES (UML) for Traceability



Moon Kun Lee
Computer Engineering
Chonbuk National University

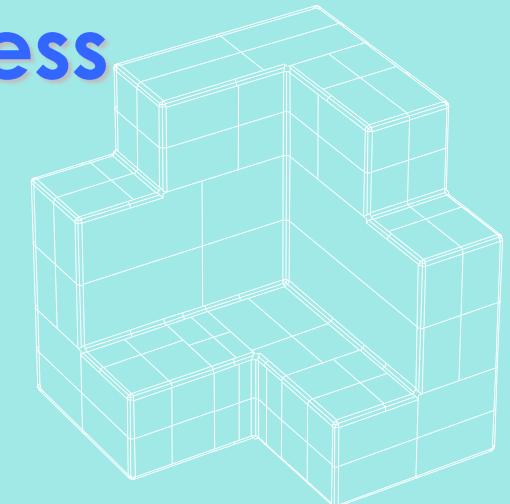
Outline

- ❖ ROPES
- ❖ Cooking System Example

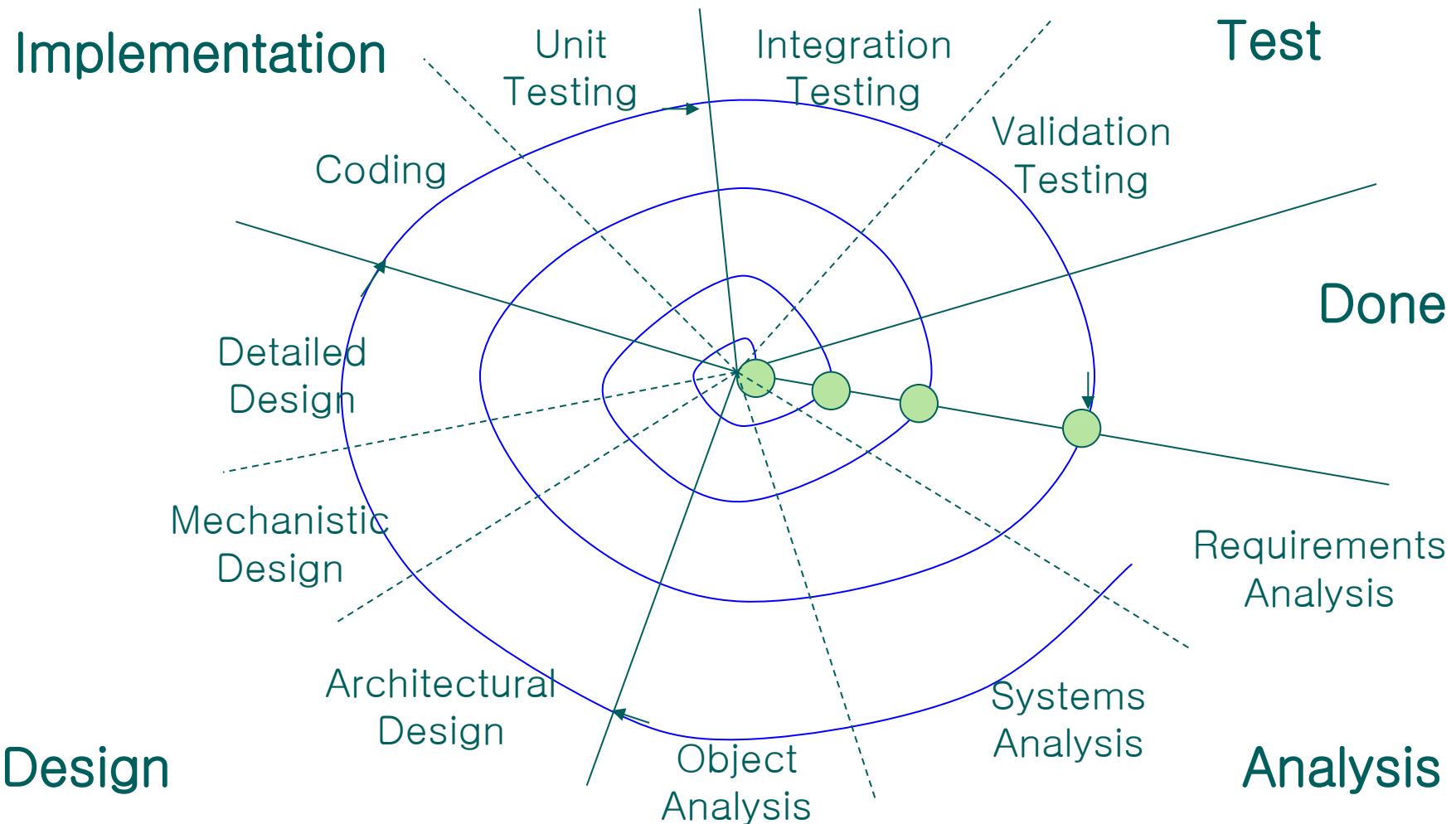
ROPEs

(Rapid Object-Oriented Process
for Embedded Systems)

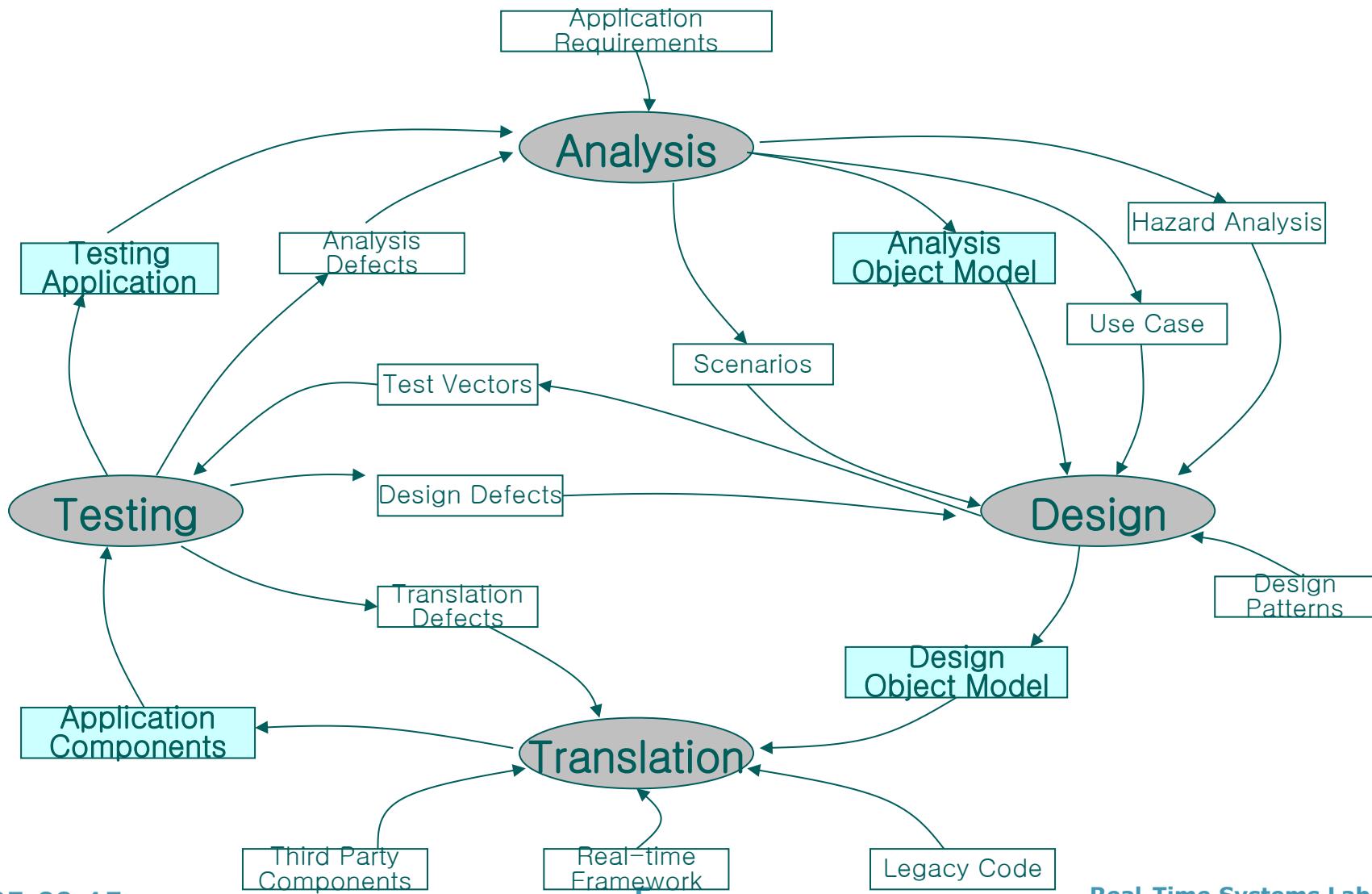
Development Phase
Artifacts
Dependencies



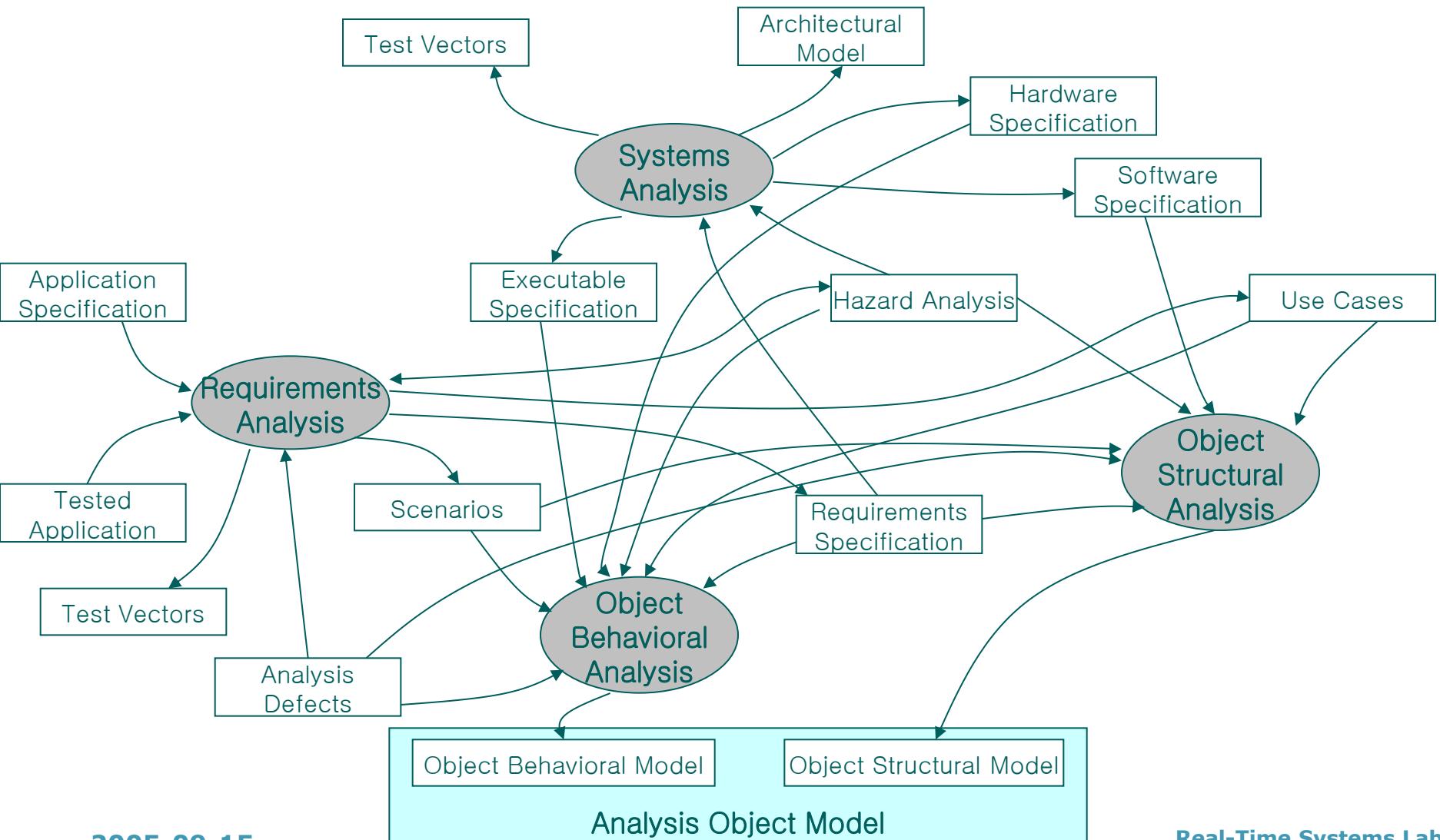
ROPEs Process



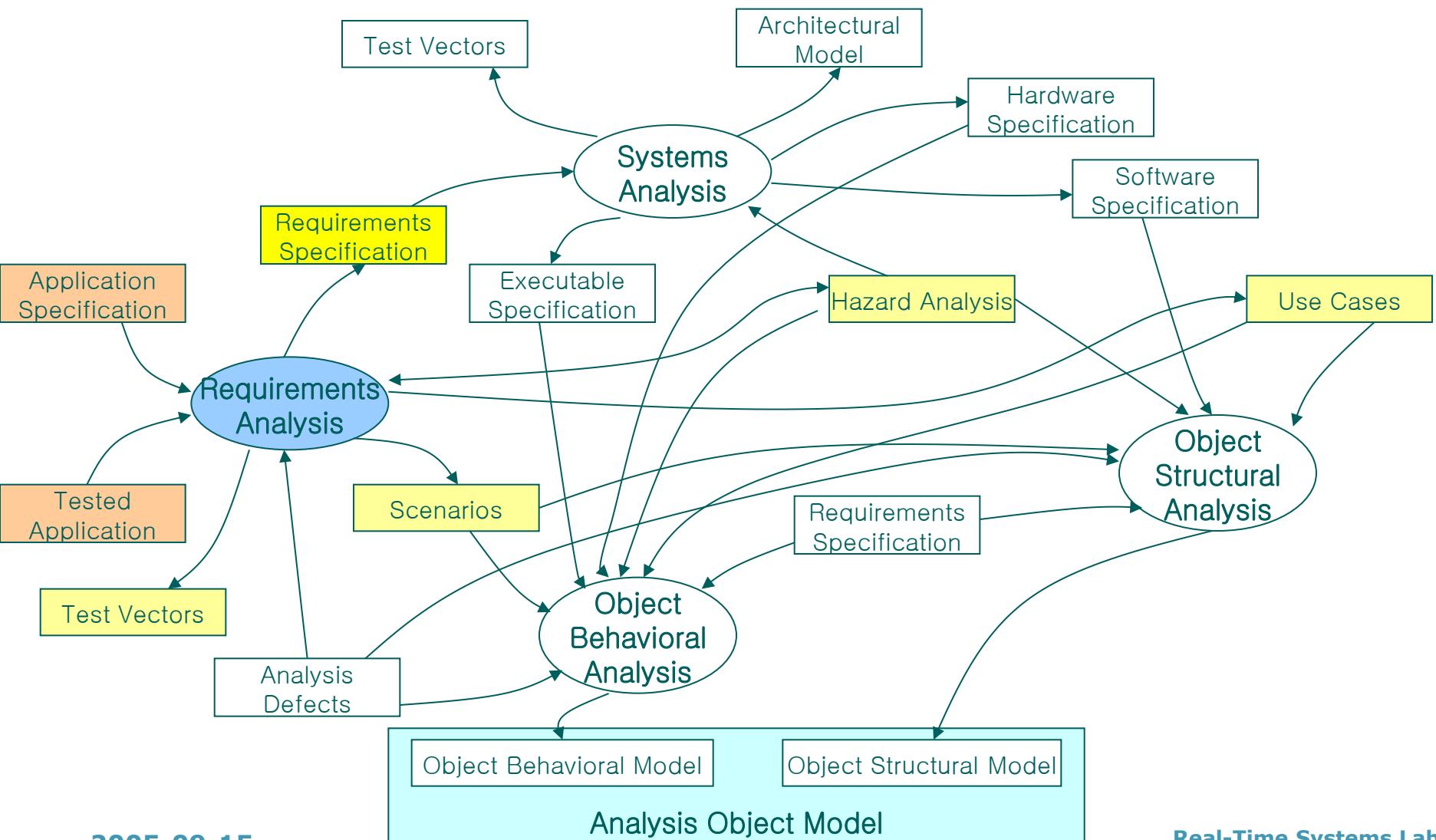
ROPS Process Artifacts



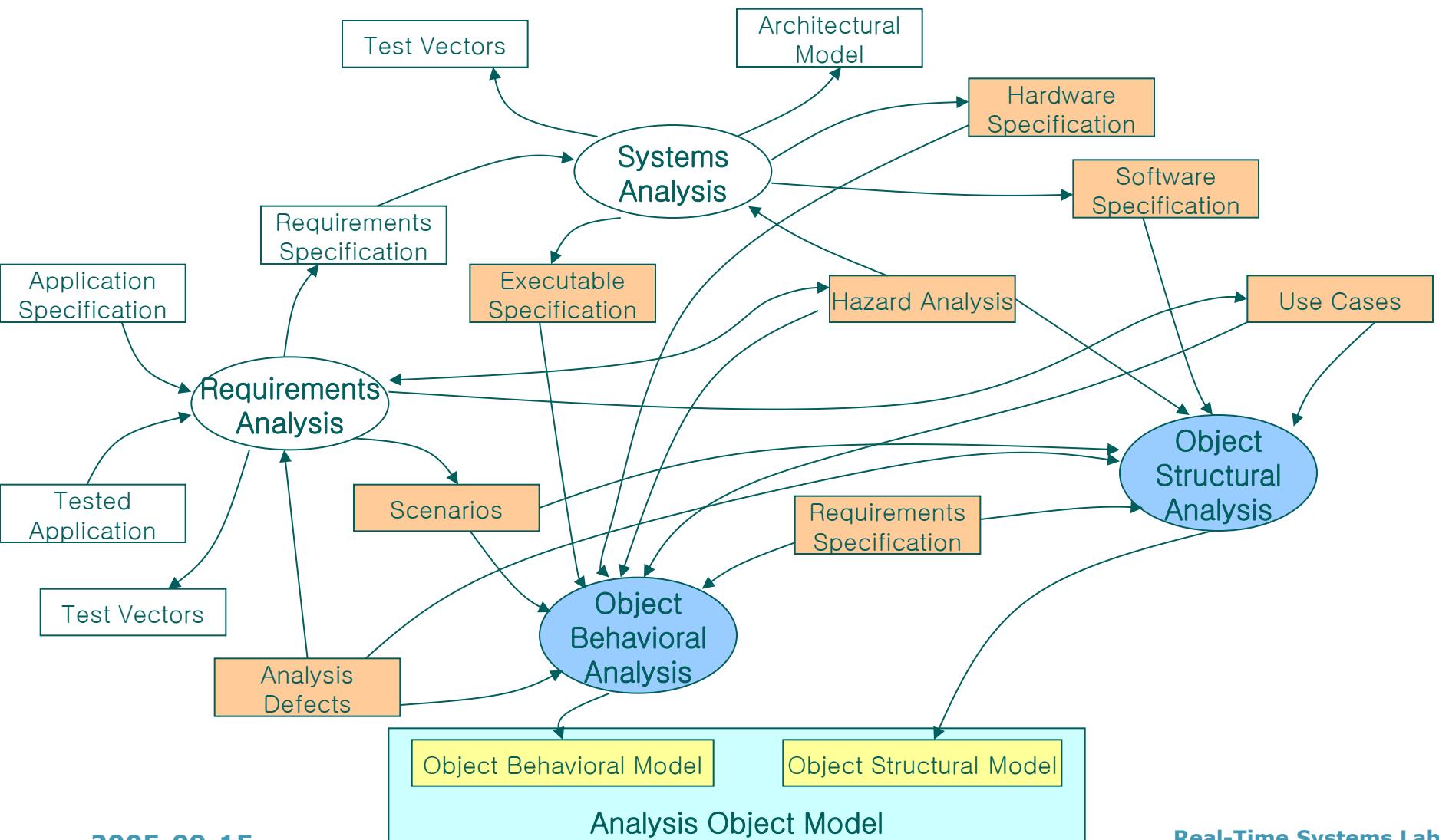
ROPEs Analysis Model Artifacts



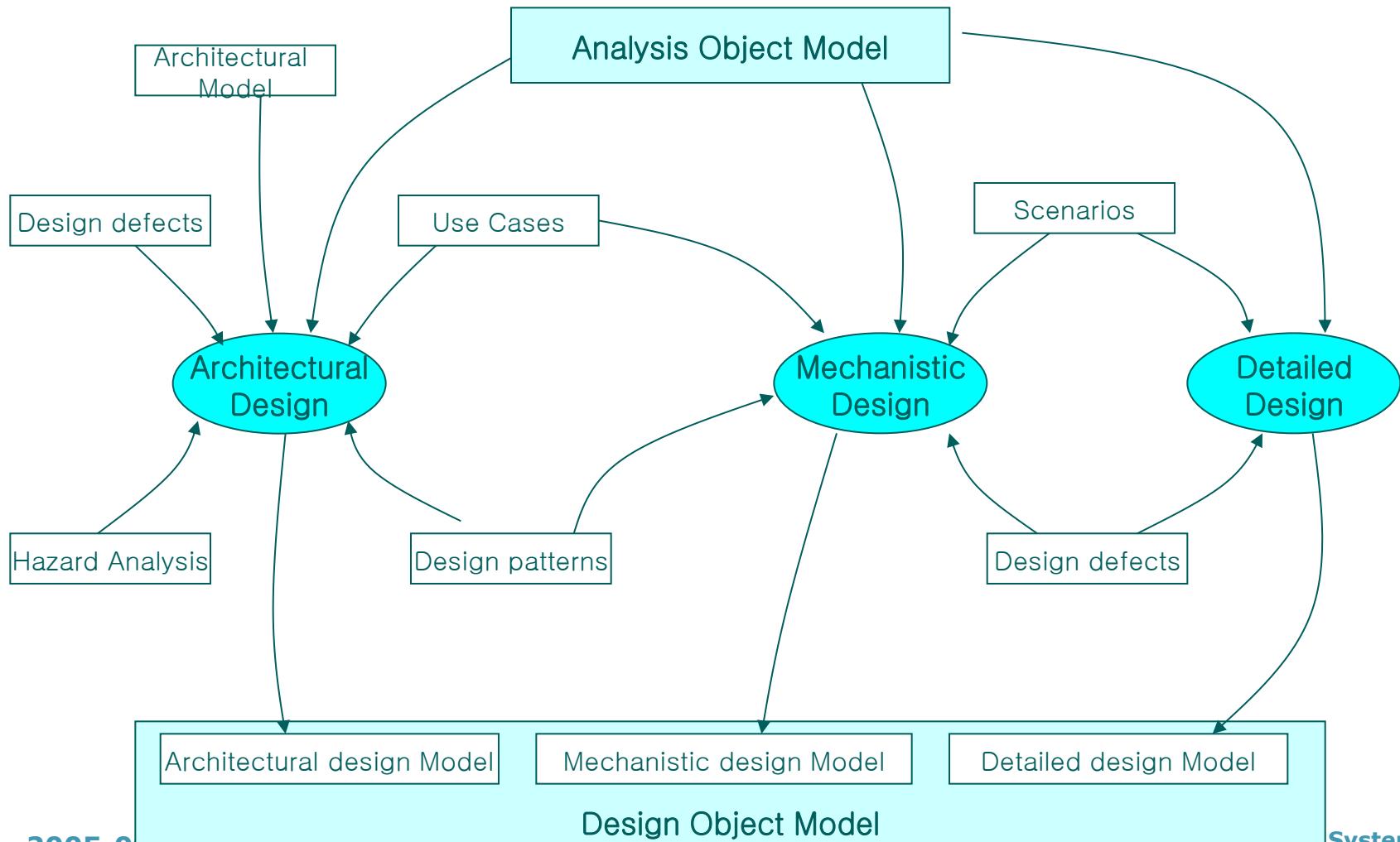
Requirements Analysis



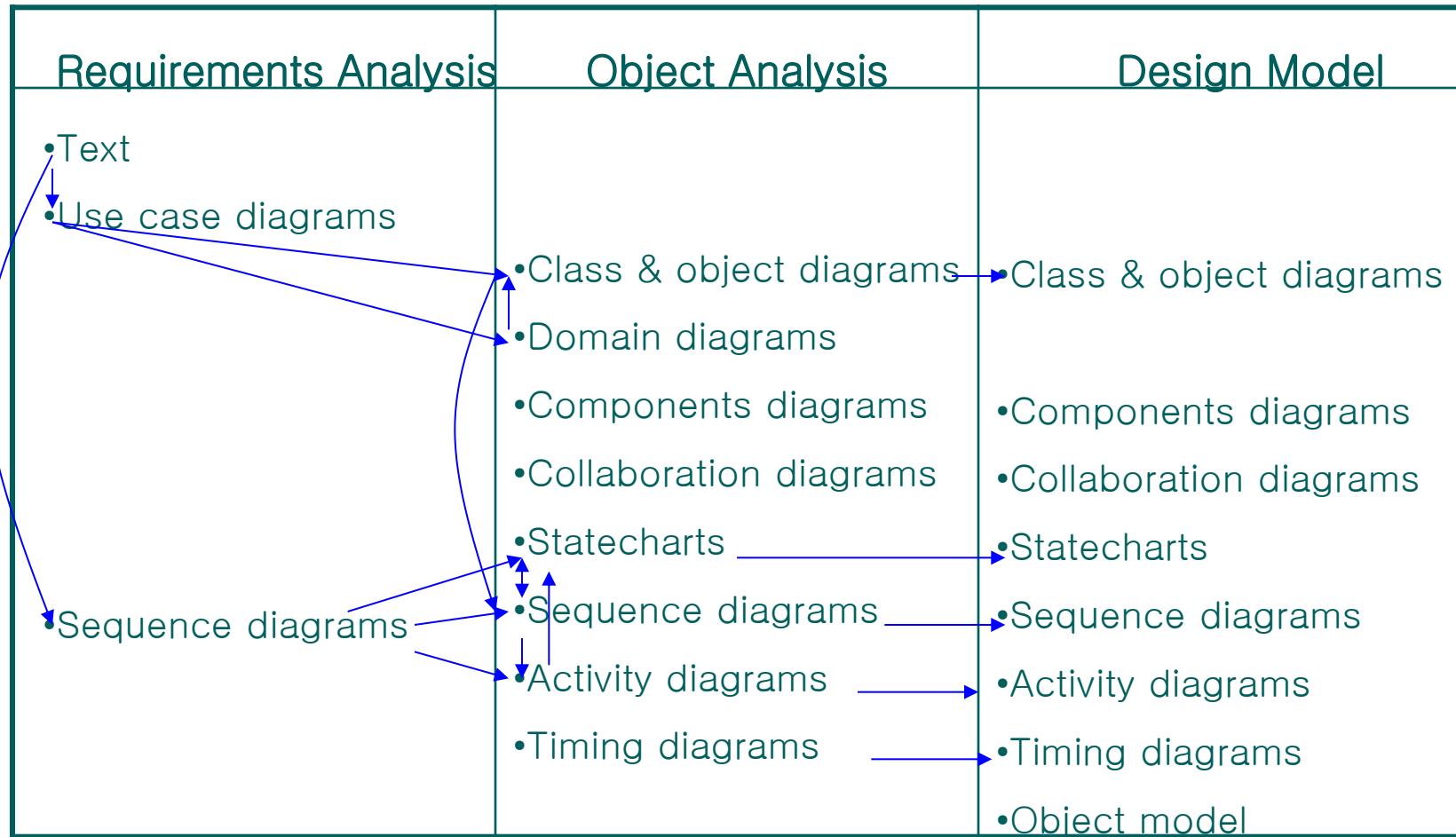
Object Structural/Behavioral Analysis



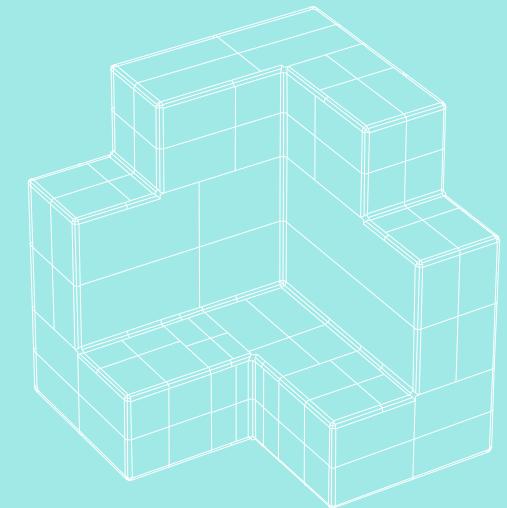
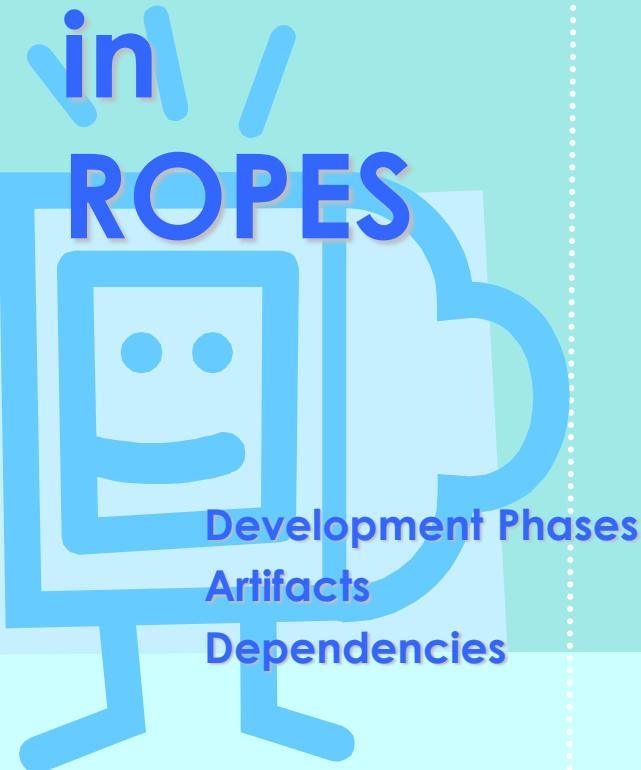
ROPS Design Model Artifacts



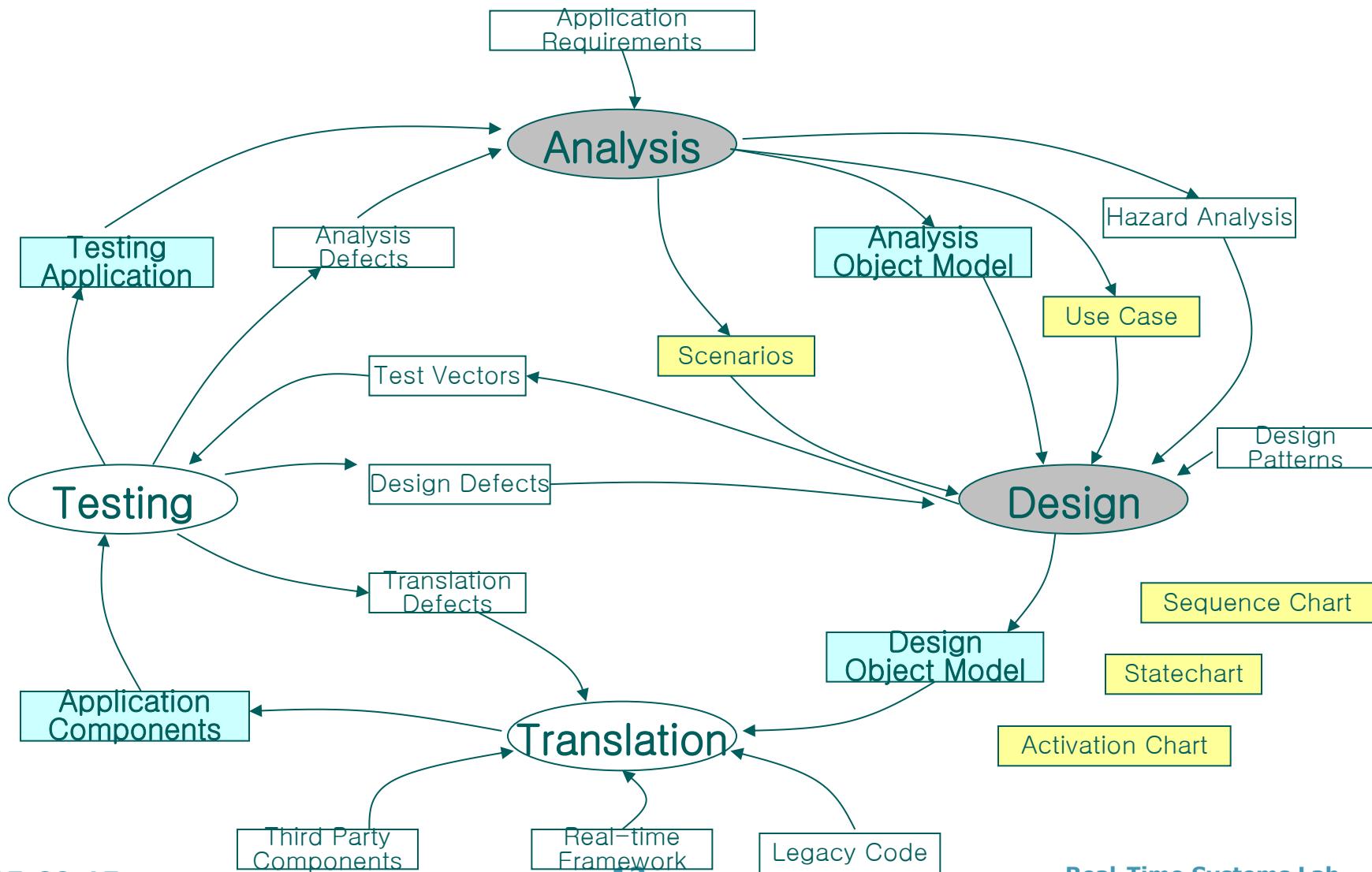
Artifacts among Phases



Cooking Example



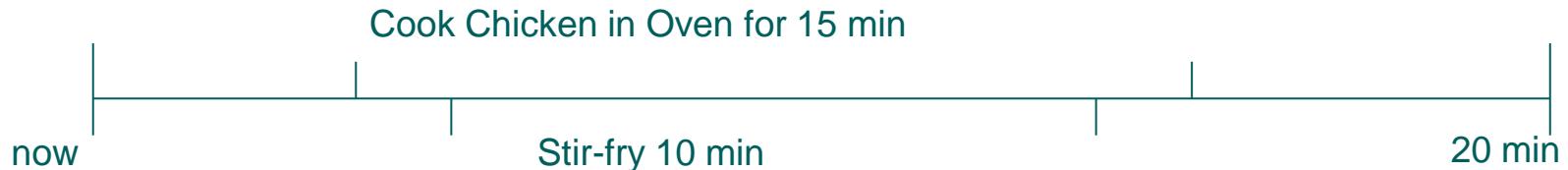
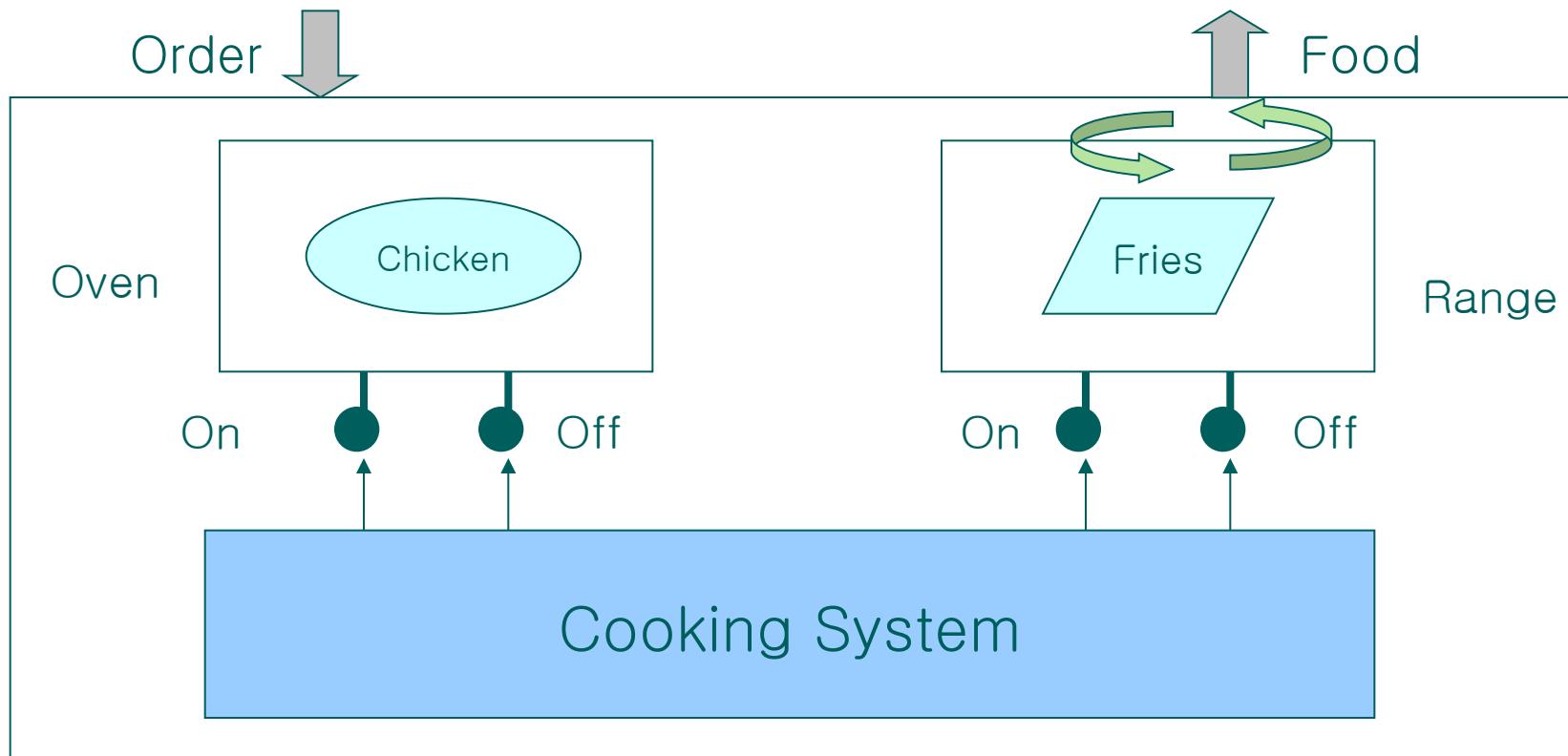
Process View



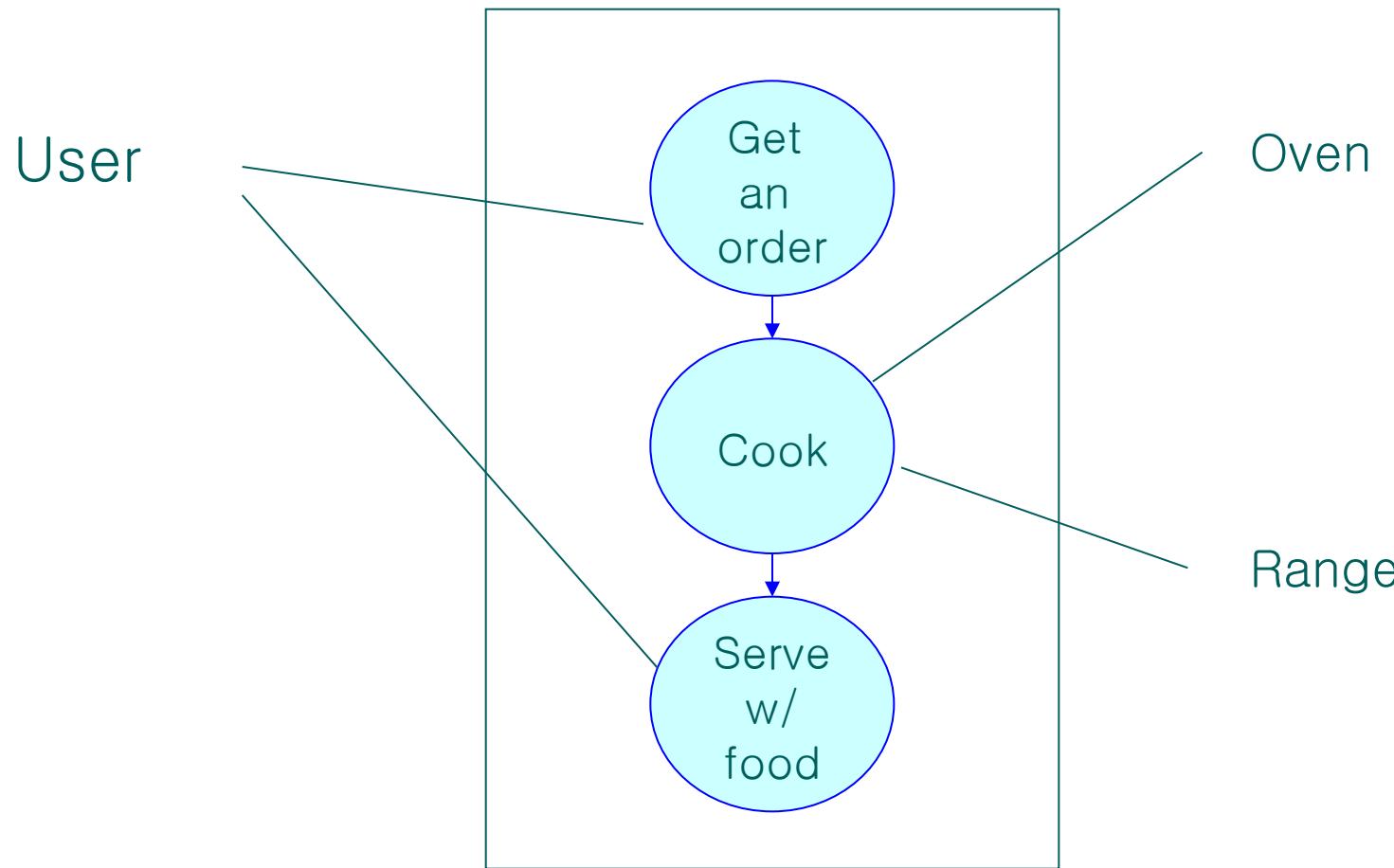
Requirements

1. The following is a list of the requirements for a cooking system.
2. When a customer orders, the system provides him/her with a piece of chicken and a bag of French fries in 20 minutes.
3. Cooking the chicken takes 15 minutes in an oven.
4. Frying takes 10 minutes in a range.
5. Cooking and frying are continuous jobs.
6. Fries should be stirred for 10 seconds in every 40 seconds while being fried.
7. Frying should start after cooking starts and end before cooking ends.

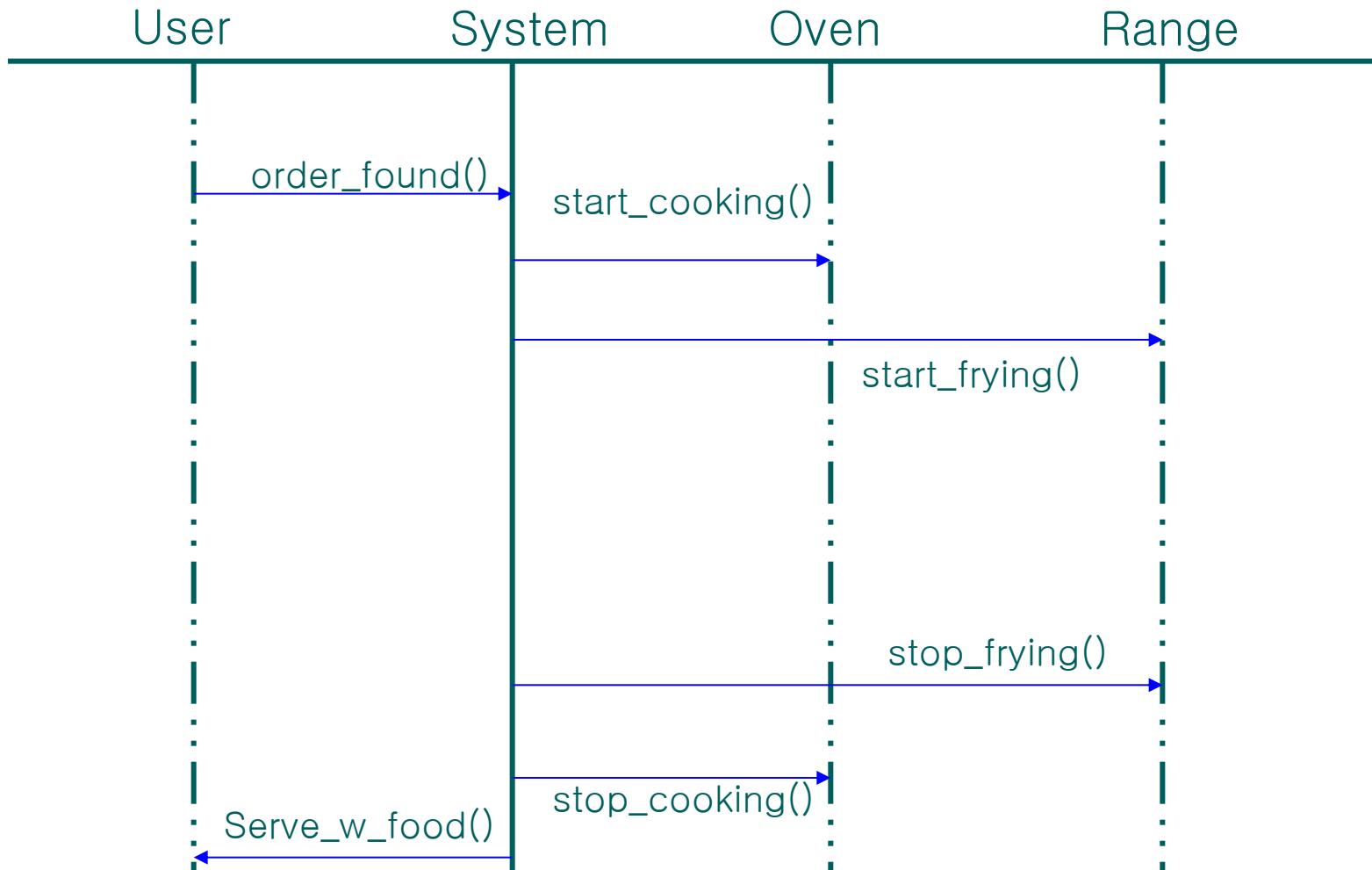
System View



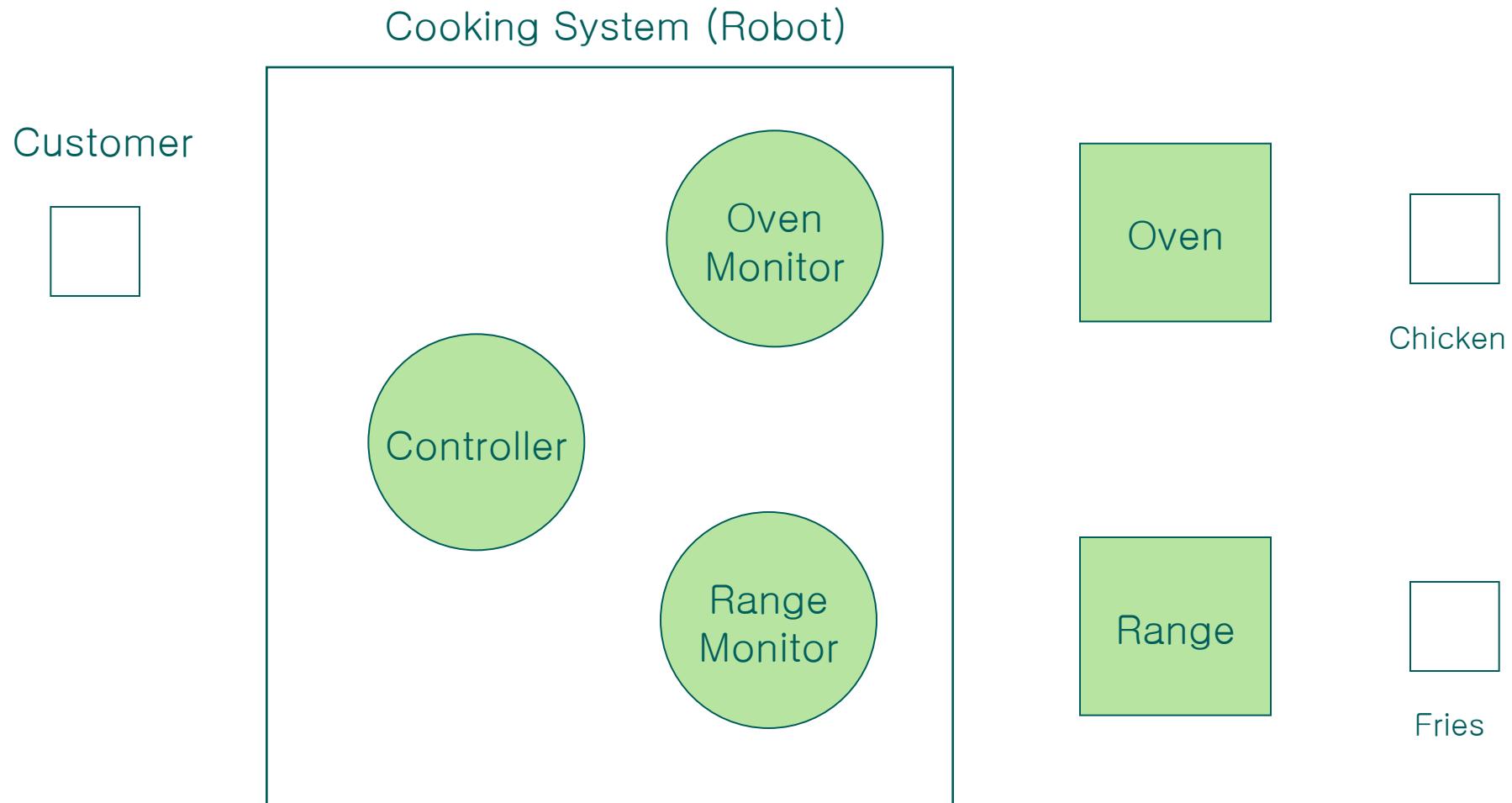
Use-Case



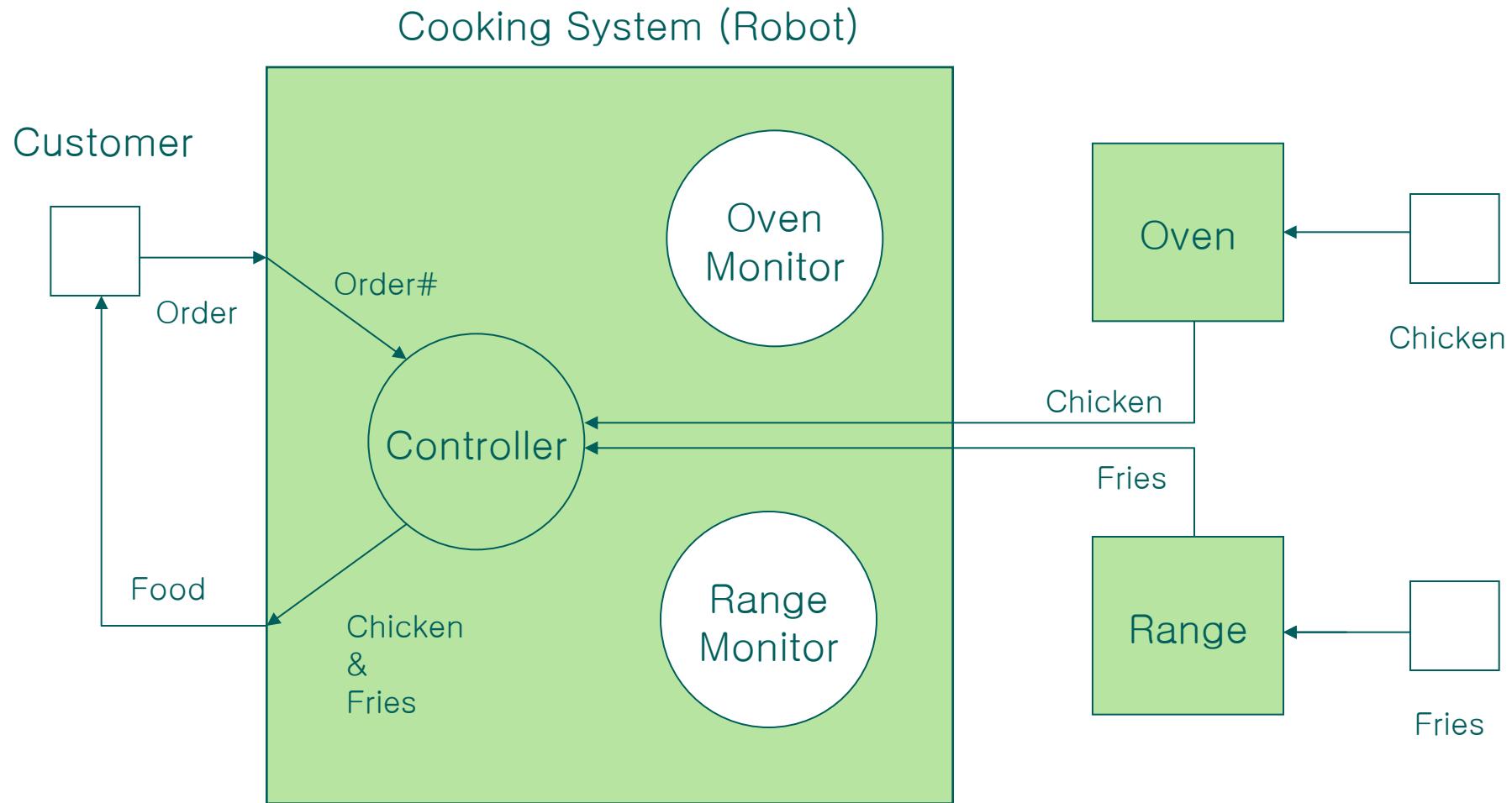
Scenario



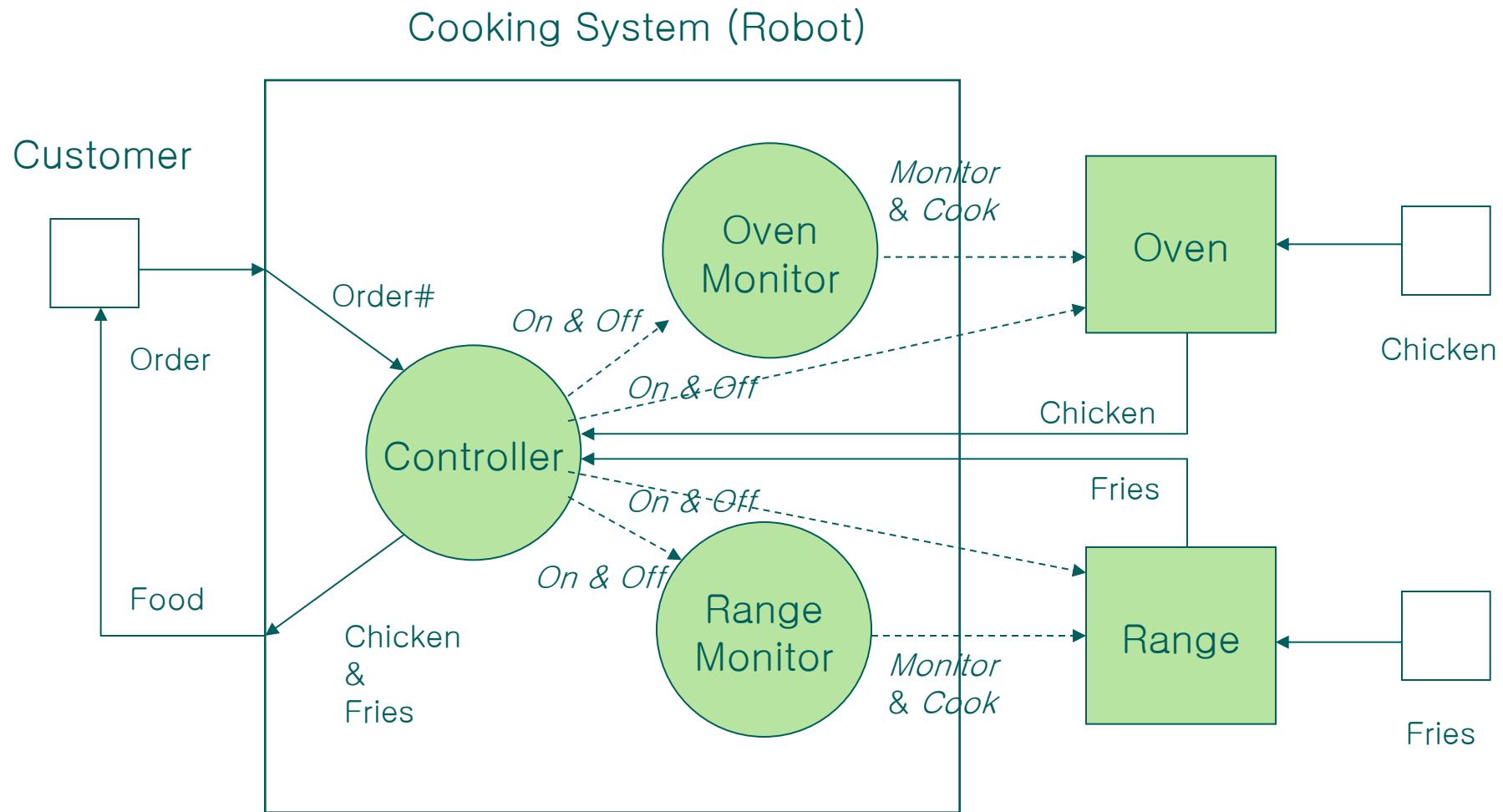
Structural



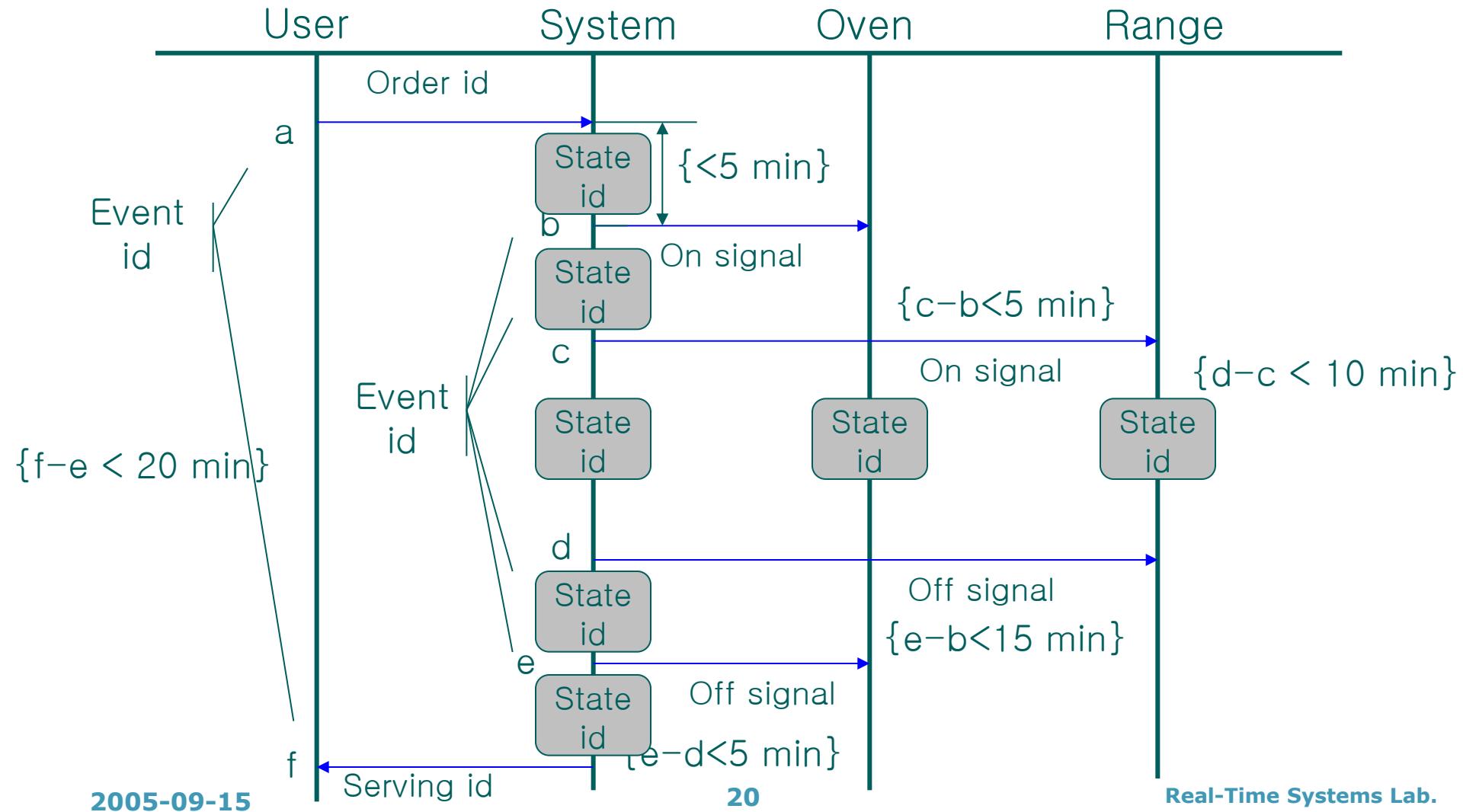
Functional



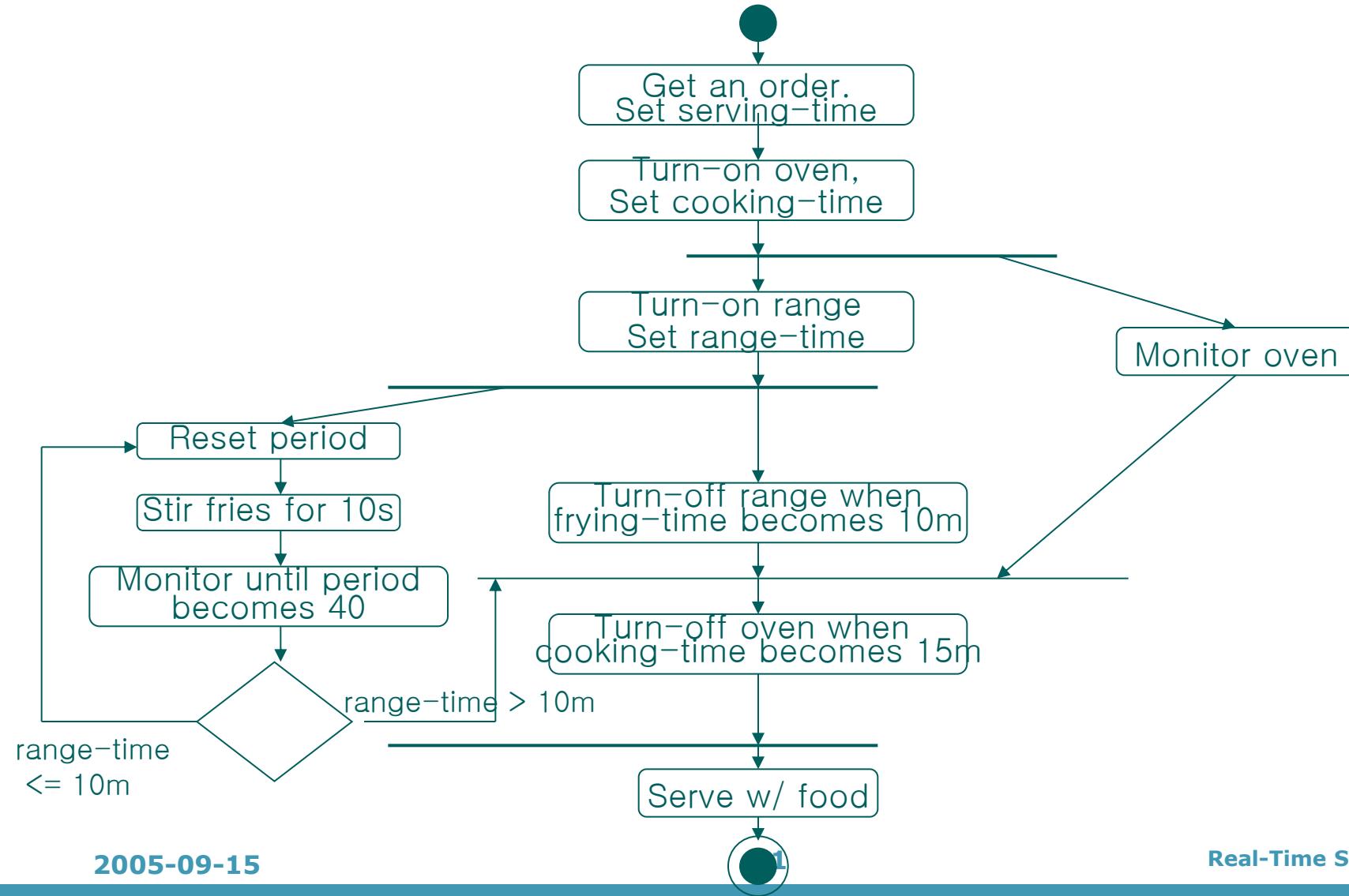
Control Flow Diagram



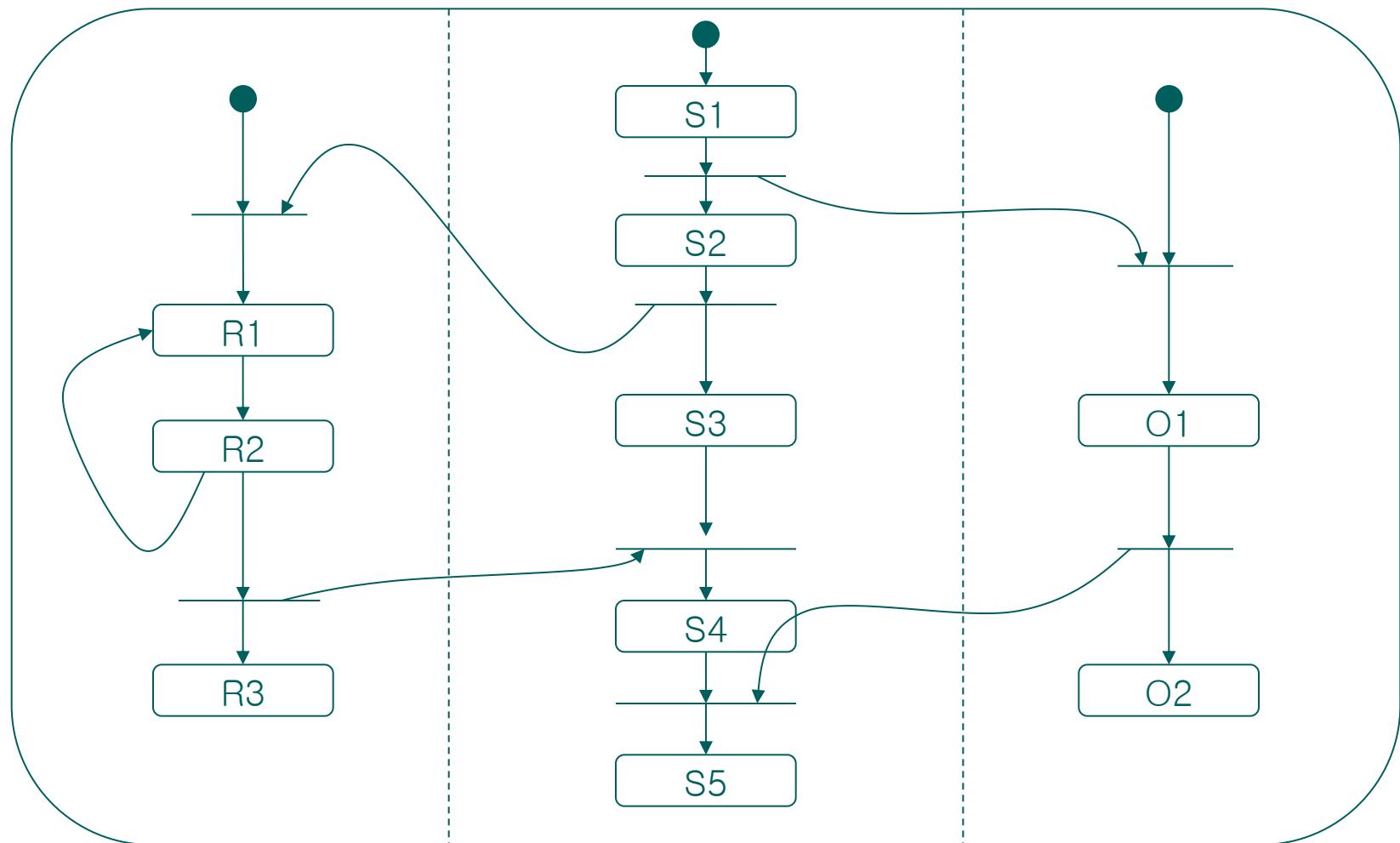
Sequence Chart



Activity Chart



Statechart



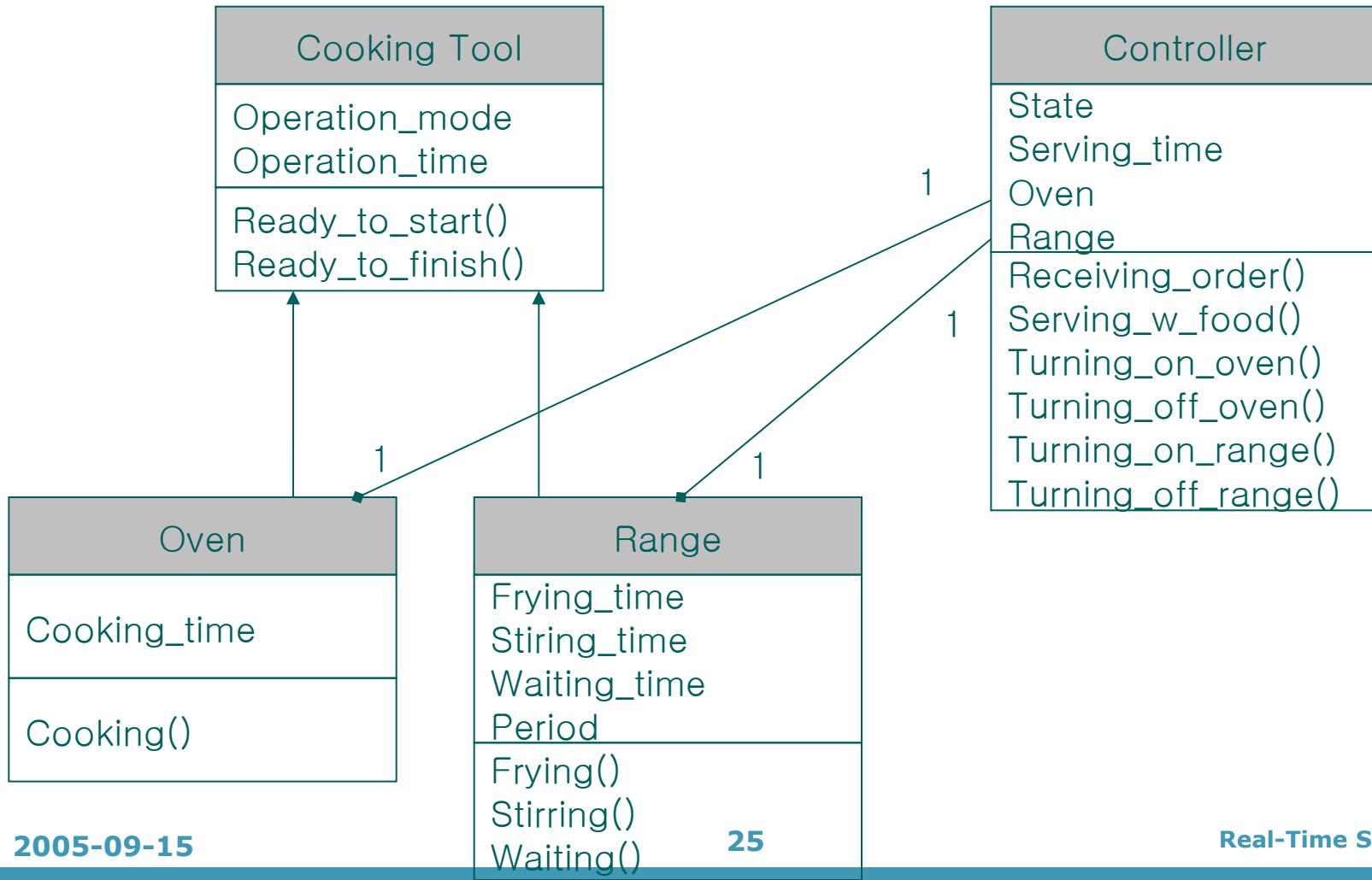
Object Identification

1. The following is a list of the requirements for a cooking system.
2. When a customer orders, the system provides him/her with a piece of chicken and a bag of French fries in 20 minutes.
3. Cooking the chicken takes 15 minutes in an oven.
4. Frying takes 10 minutes in a range.
5. Cooking and frying are continuous jobs.
6. Fries should be stirred for 10 seconds in every 40 seconds while being fried.
7. Frying should start after cooking starts and end before cooking ends.

Object Property Identification

Object	Attributes		Methods	
	Explicit	Implicit	Explicit	Implicit
Controller		Ordering_time Serving_time Serving_duration Cooking_time Frying_tile Order_id	Cooking() Frying() Stirring()	
Oven		Operation_mode Operation_time Cooking_time	Cooking()	
Range		Operation_mode Operation_time Frying_time Period Waiting_time	Frying()	Stirring() Waiting()

Class Diagram



Event/action Identification

1. The following is a list of the requirements for a cooking system.
2. When a customer **orders**, the system **provides** him/her with a piece of chicken and a bag of French fries in 20 minutes.
3. **Cooking** the chicken takes 15 minutes in an oven.
4. **Frying** takes 10 minutes in a range.
5. Cooking and frying are continuous jobs.
6. Fries should be **stirred** for 10 seconds in every 40 seconds while being fried.
7. Frying should **start** after cooking **starts** and **end** before cooking **ends**.

Event Property

Object	Attributes		Methods	
	Event	Property	Event	Property
Controller	Ordering_time Serving_time Serving_duration Cooking_time Frying_time Order_id	t(order) t(serving) t(serving)-t(order) $c(\text{cooking_time})=15\text{m}$ $c(\text{fryng_time})=10\text{m}$	Cooking() Frying() Stirring()	
Oven	Operation_mode Operation_time Cooking_time	$c(\text{cooking_time})=15\text{m}$	Cooking()	
Range	Operation_mode Operation_time Frying_time Stirring_time Period Waiting_time	$c(\text{fryng_time})=10\text{m}$ $c(\text{Stirring_time})=10\text{s}$ $c(\text{Period})=40\text{s}$	Frying() Stirring() Waiting()	

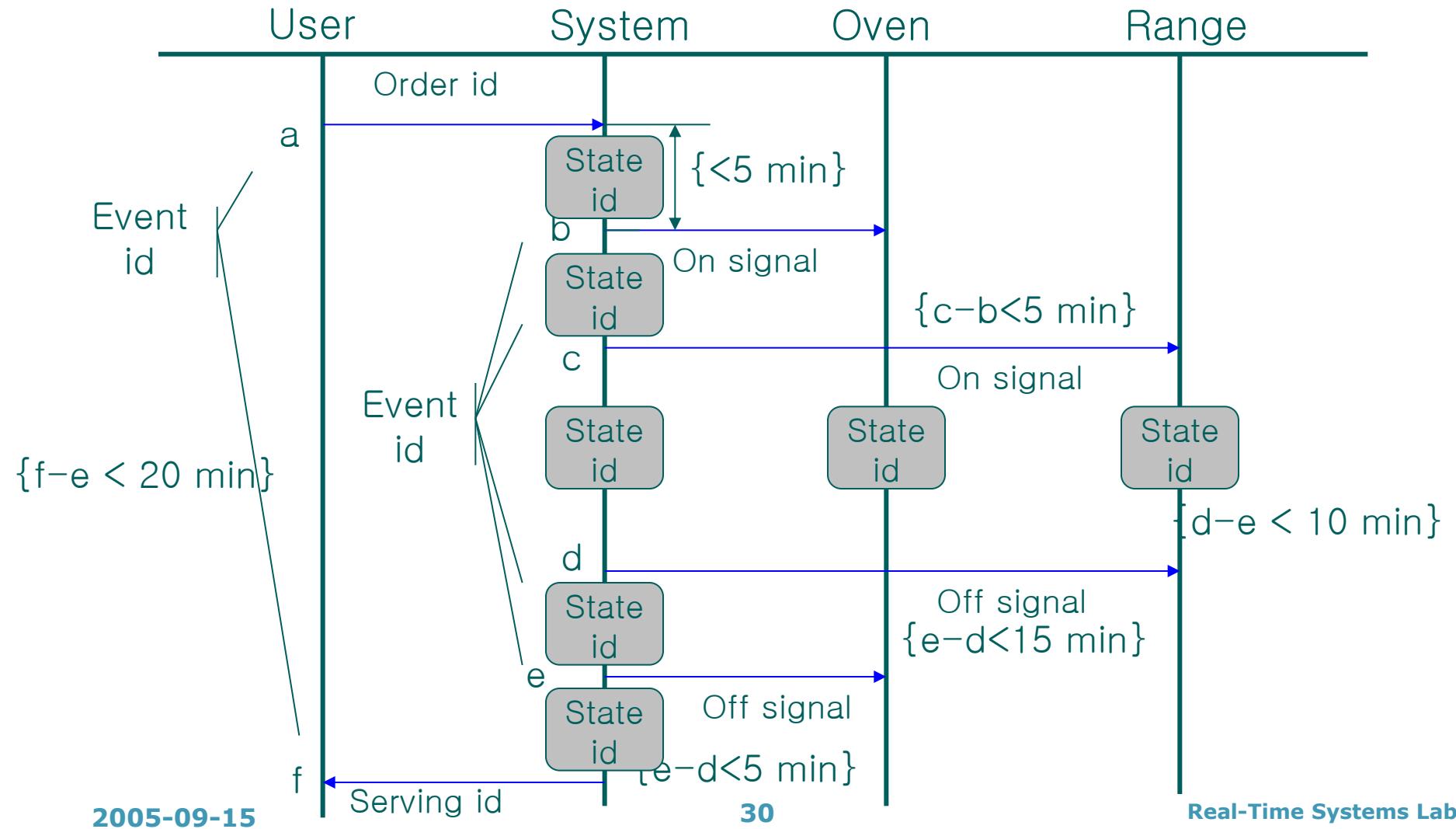
Timing/Ordering Requirements

1. The following is a list of the requirements for a cooking system.
2. When a customer **orders**, the system **provides** him/her with a piece of chicken and a bag of French fries in 20 minutes.
R2-1
3. **Cooking** the chicken takes 15 minutes in an oven.
4. **Frying** takes 10 minutes in a range.
R4-1 R3-1
5. Cooking and frying are continuous jobs.
6. Fries should be **stirred** for 10 seconds in every 40 seconds while being fried.
7. Frying should **start** after cooking **starts** and **end** before frying **ends**.

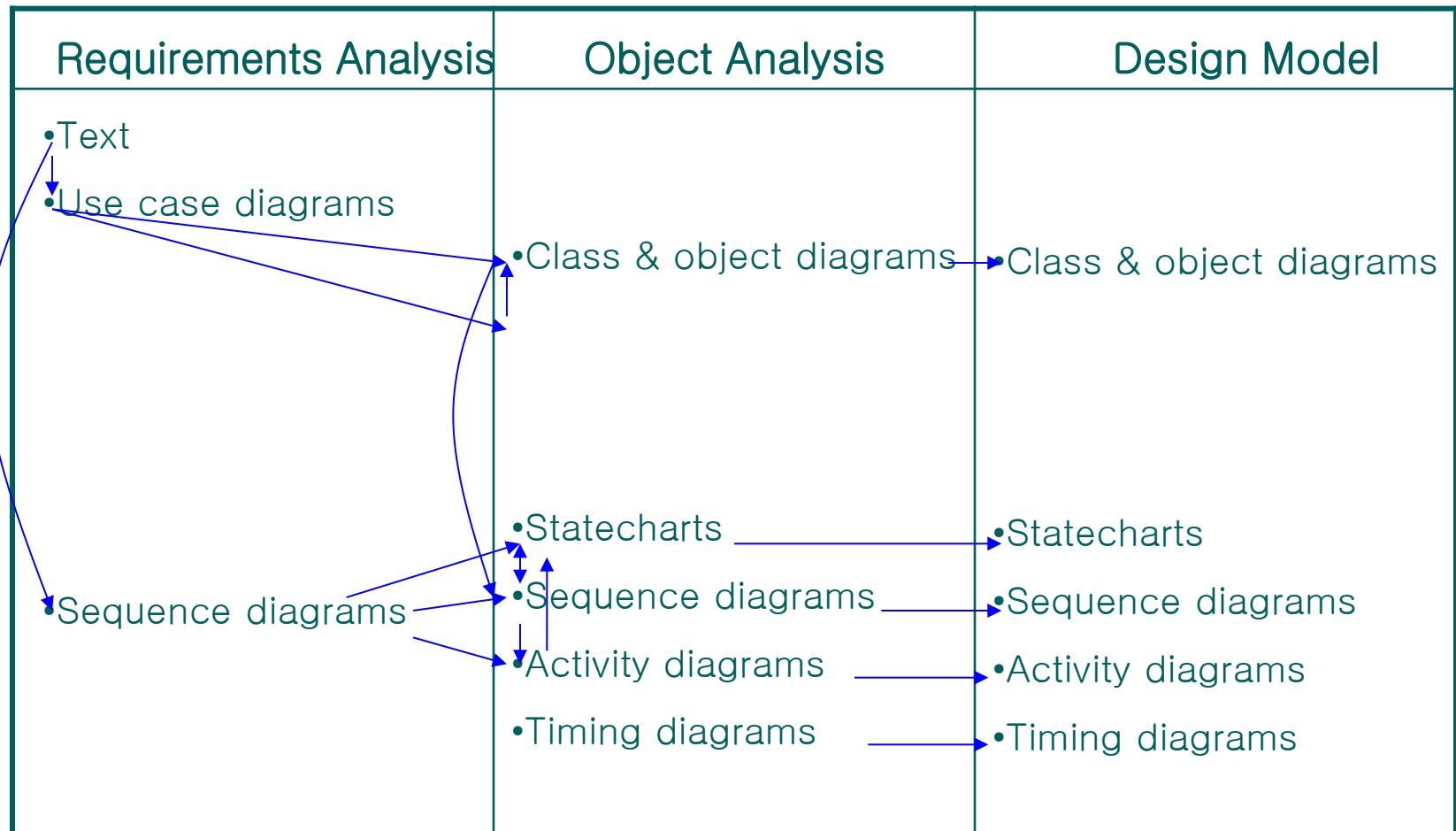
Relation Requirements Properties

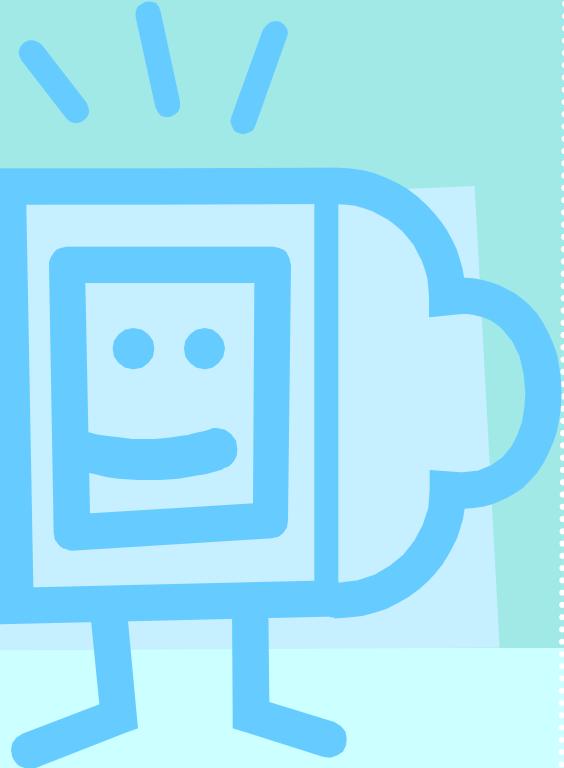
Req	Properties
R2-1	$t(\text{controller}'\text{order}) - t(\text{controller}'\text{serve}) < 20m$
R3-1	$t(\text{oven}'\text{operation} == \text{off}) - t(\text{oven}'\text{operation} == \text{on}) < 15m$
R4-1	$t(\text{range}'\text{operation} == \text{off}) - t(\text{range}'\text{operation} == \text{on}) < 10m$
R7-1	$t(\text{oven}'\text{operation} == \text{on}) - t(\text{range}'\text{operation} == \text{on}) \geq 0s$
R7-2	$t(\text{oven}'\text{operation} == \text{off}) - t(\text{range}'\text{operation} == \text{off}) \leq 0s$

Sequence Diagram

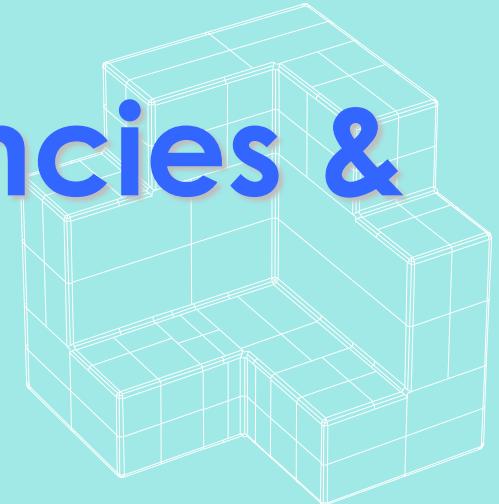


Relations among Artifacts





Dependencies & Relations



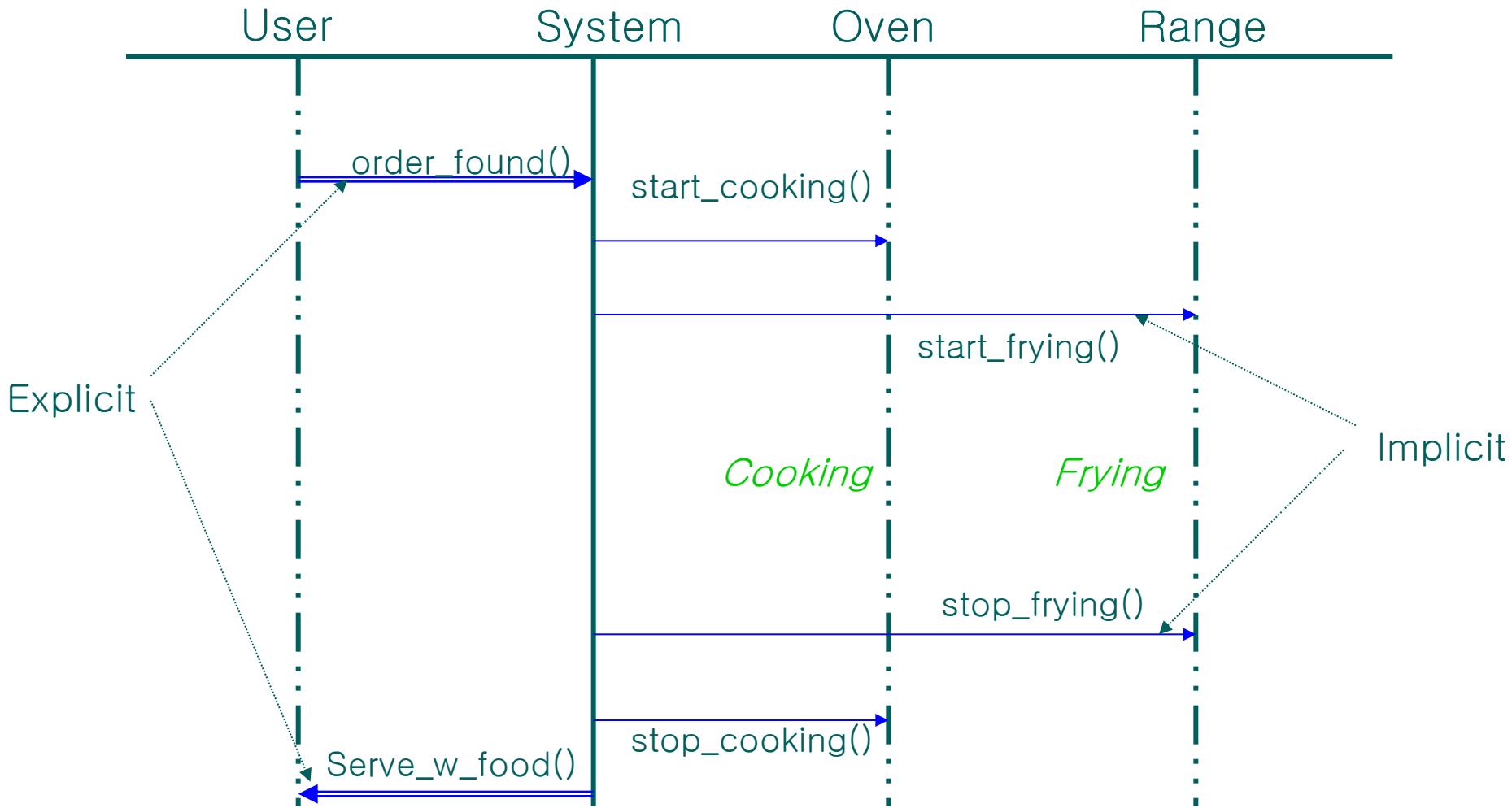
Dependency

- ❖ System components
- ❖ Interaction/event
- ❖ Ordering/timing
- ❖ Explicit/implicit
- ❖ Actualization

Object/Interaction/ordering

1. The following is a list of the requirements for a cooking system.
2. When a customer orders, the system provides him/her with a piece of chicken and a bag of French fries in 20 minutes.
3. *Cooking* the chicken takes 15 minutes in an oven.
4. *Frying* takes 10 minutes in a range.
5. *Cooking* and *frying* are continuous jobs.
6. Fries should be stirred for 10 seconds in every 40 seconds while being fried.
7. *Frying* should start after *cooking* starts and end before *cooking* ends.

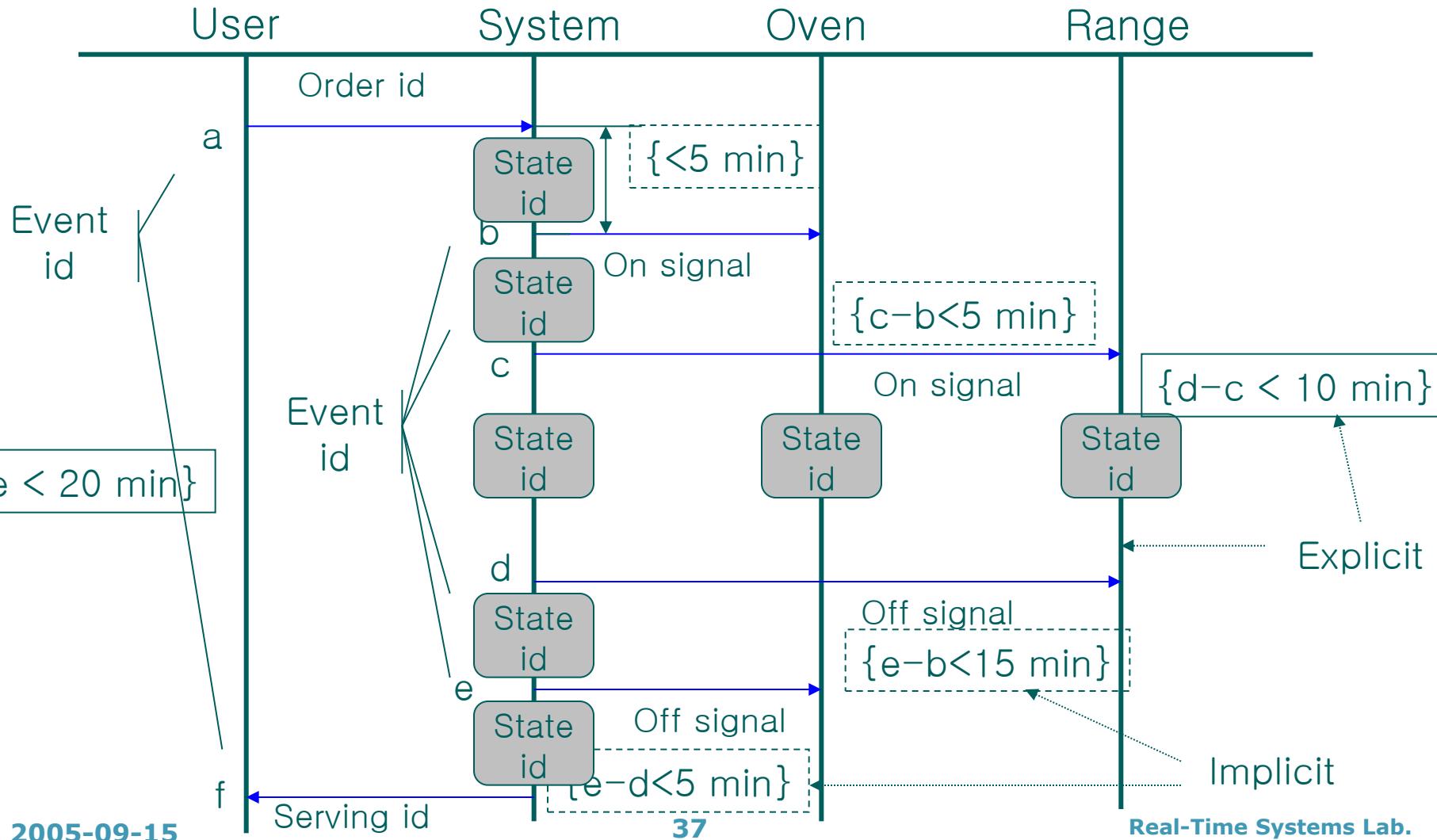
Scenario



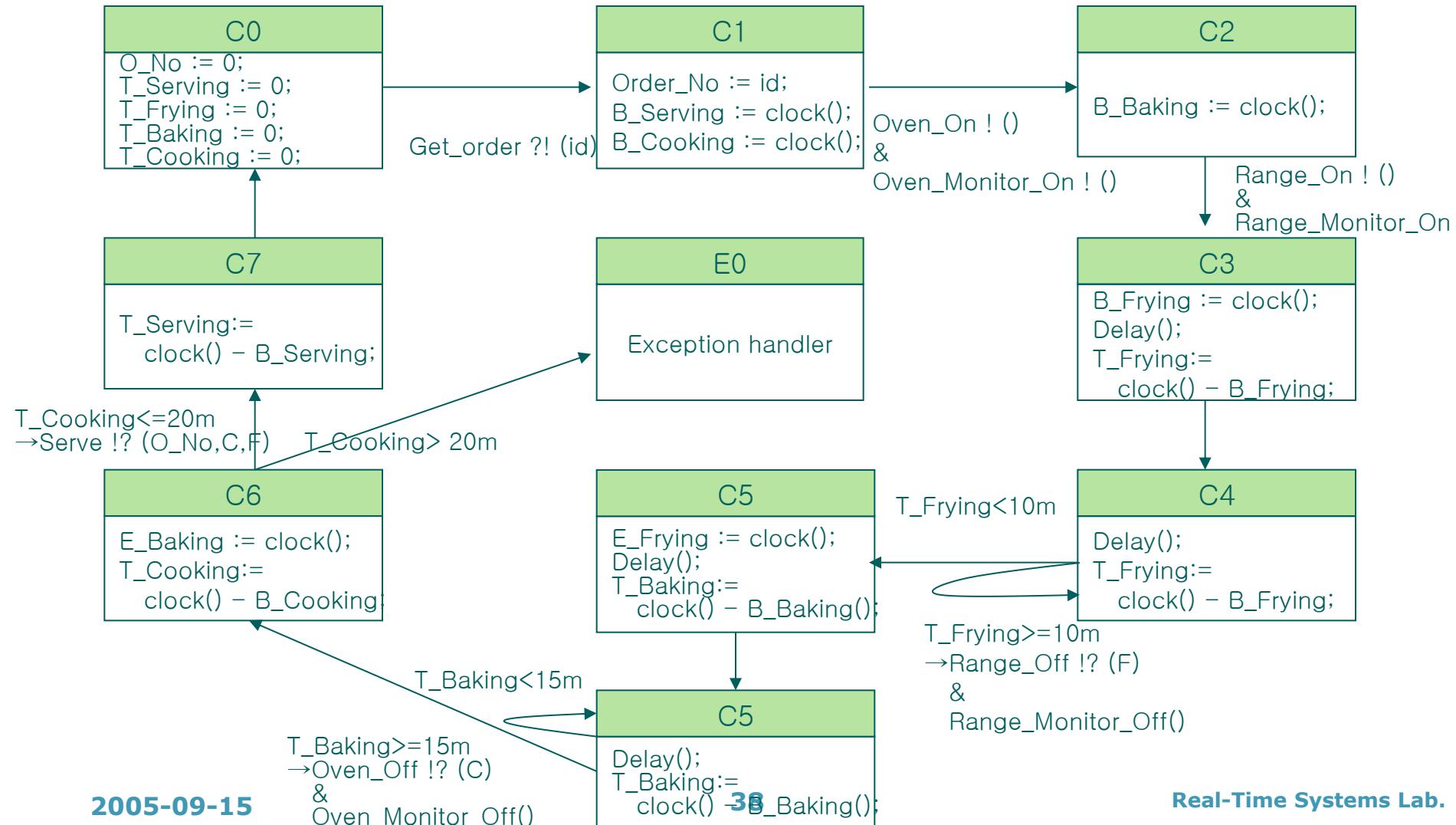
Ordering/Timing

1. The following is a list of the requirements for a cooking system.
2. When a customer orders, the system provides him/her with a piece of chicken and a bag of French fries in 20 minutes.
3. *Cooking* the chicken takes 15 minutes in an oven.
4. *Frying* takes 10 minutes in a range.
5. *Cooking* and *frying* are continuous jobs.
6. Fries should be stirred for 10 seconds in every 40 seconds while being fried.
7. *Frying* should start after *cooking* starts and end before *cooking* ends.

Sequence Chart



State Machine: Controller



SM: Introduction

- Space:

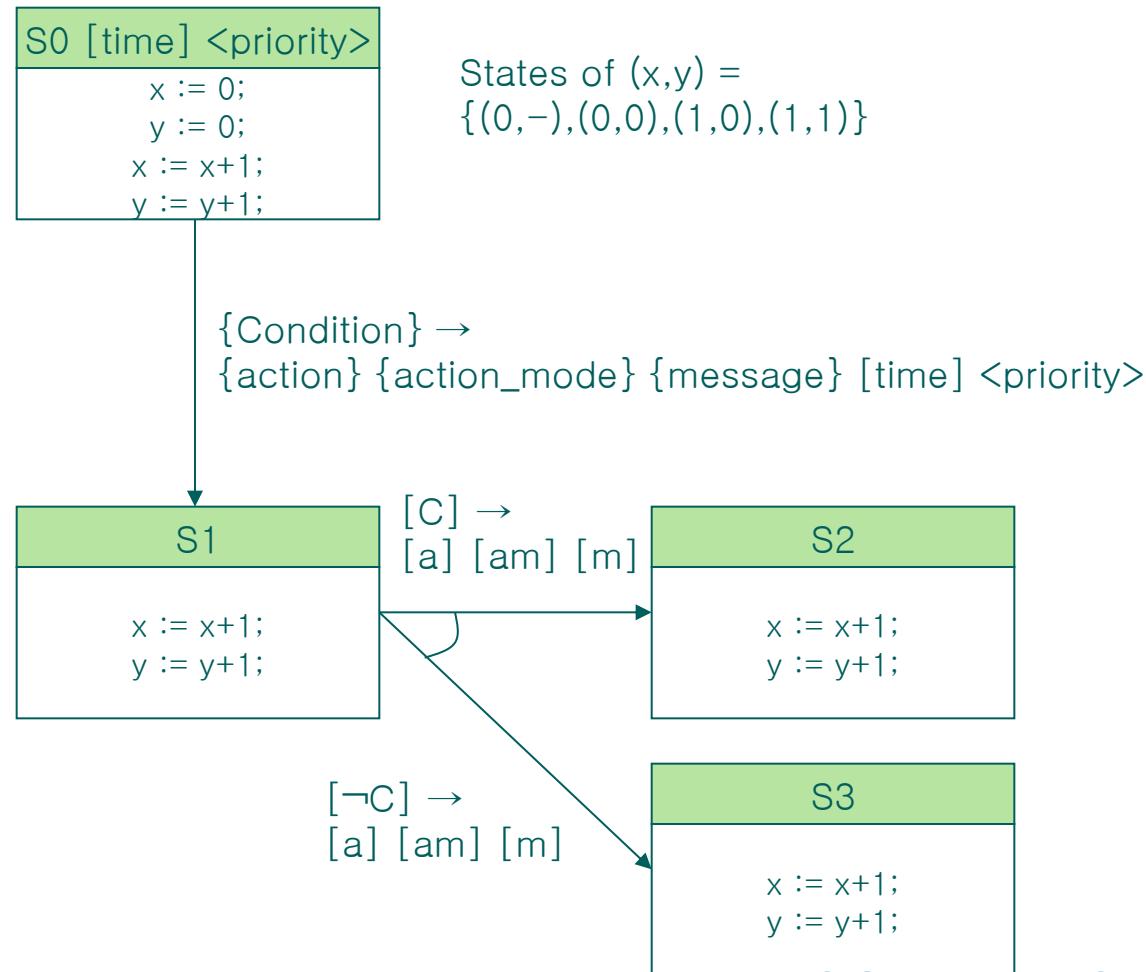
- set of states
- properties:
 - + time
 - + priority

- Transition:

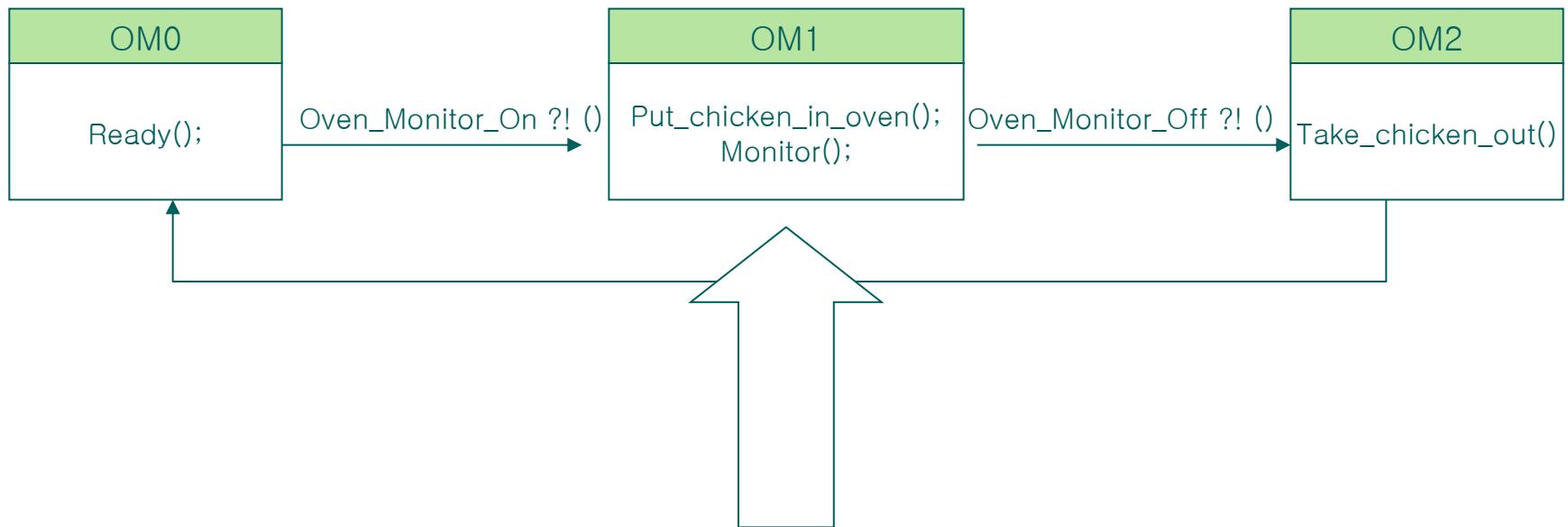
- change of space
- properties:
 - + time
 - + priority

- Action:

- mode:
 - + asynchronous:
 - send: !
 - receive: ?
 - + synchronous:
 - send: !?
 - receive: ?!

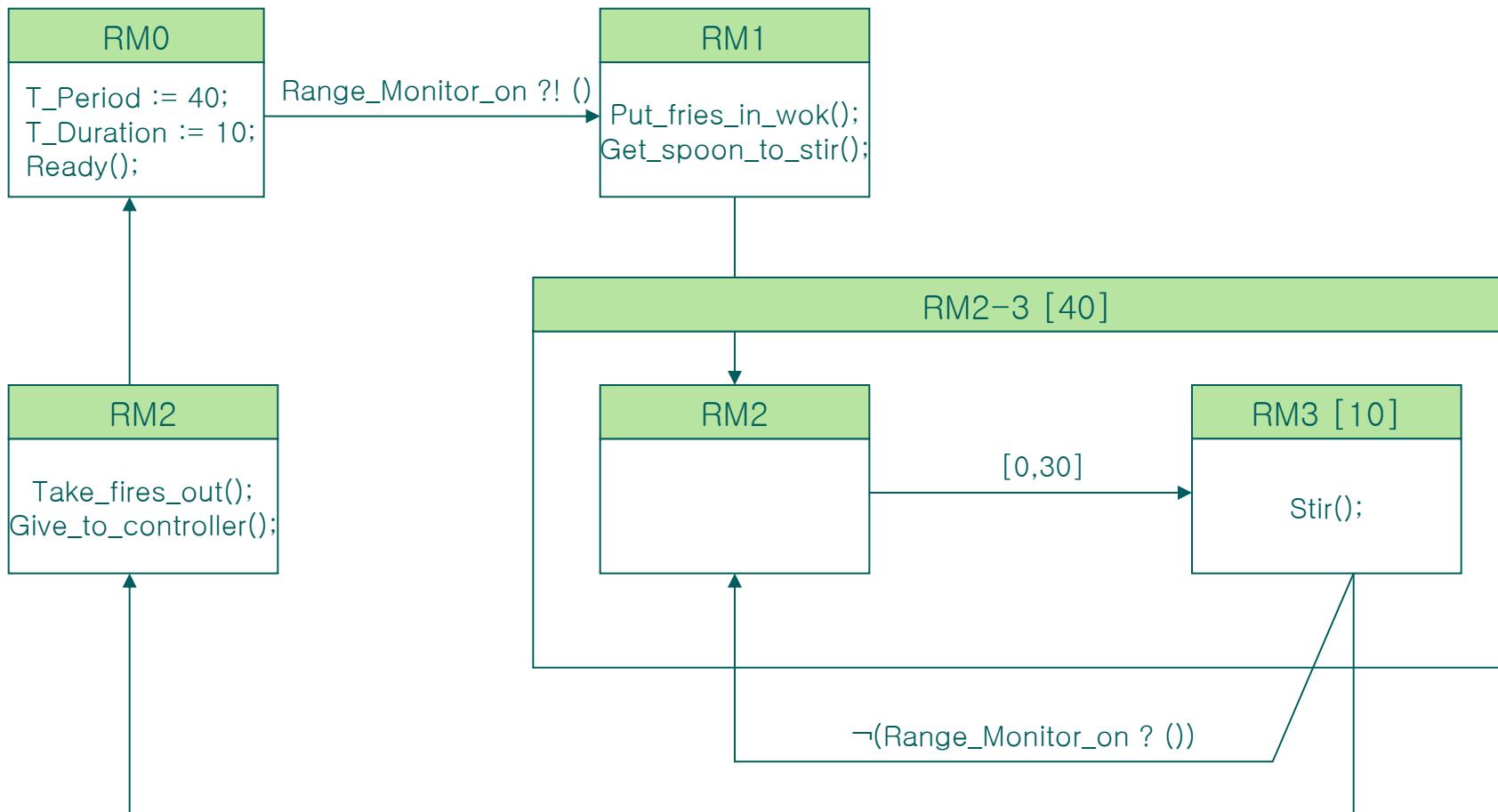


Oven Monitor

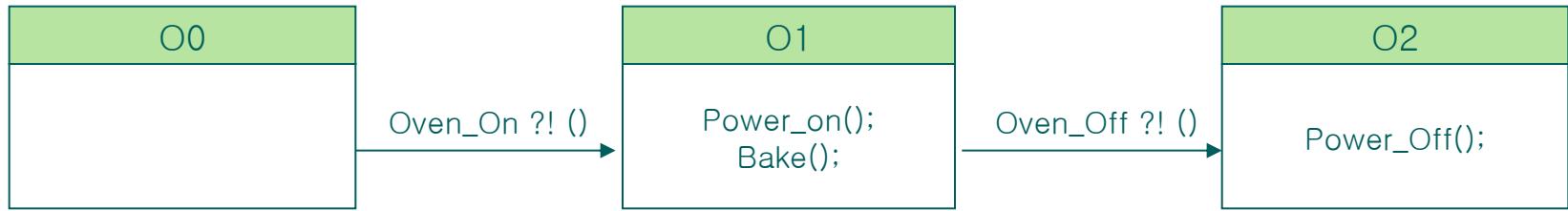


Monitoring activities can be specified,
i.e., temperature, faces, distance, etc.

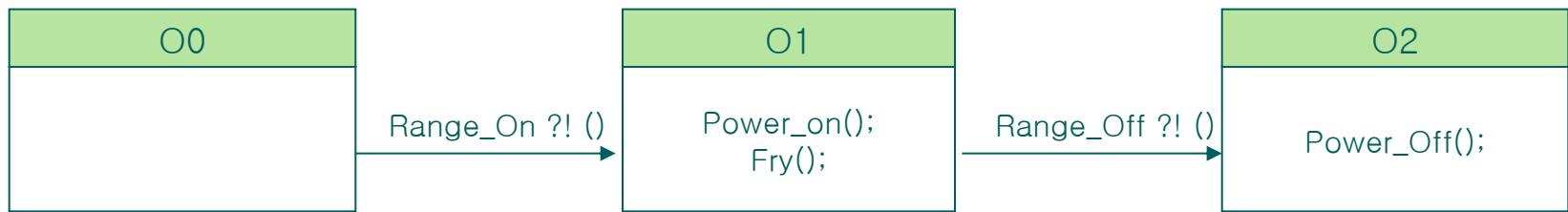
Range Monitor

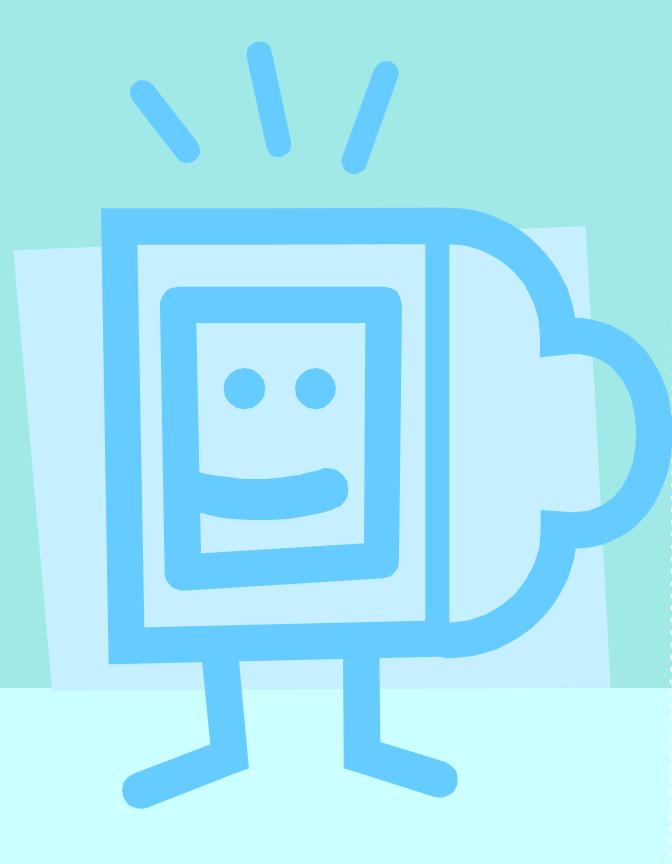


Oven

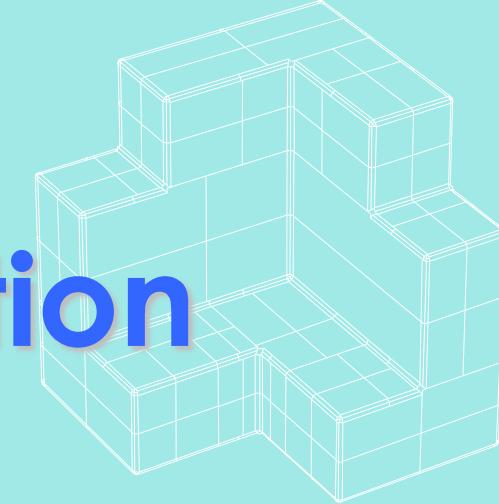


Range





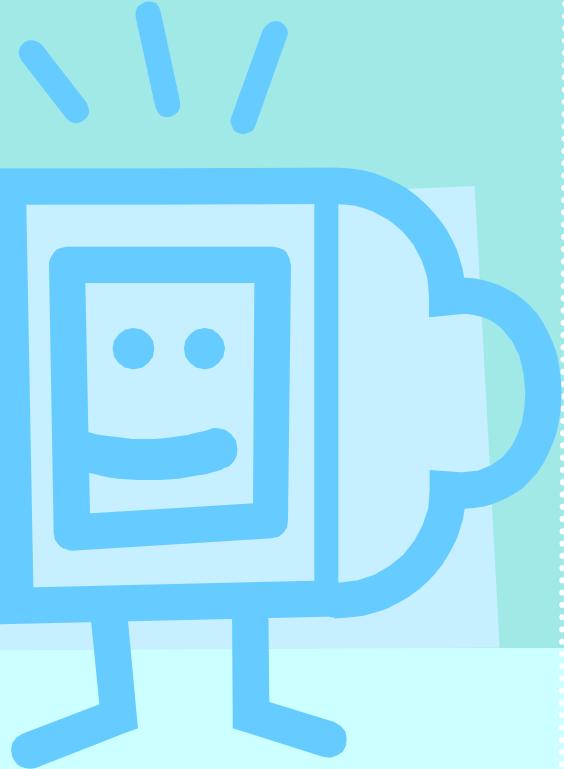
Code: Specification



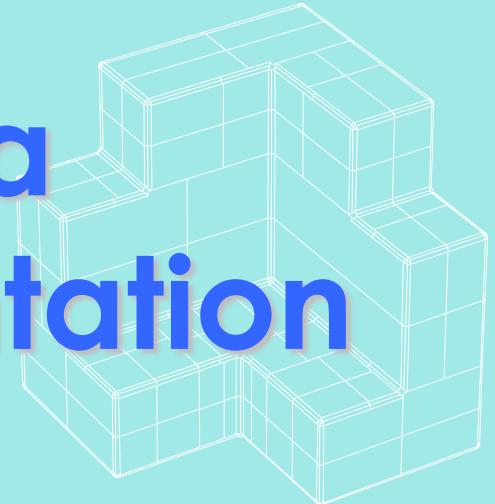
```

process cooking_robot;
call-port RangeOn [deadline 2 sec], RangeOff [deadline 2 sec];
    /* Fry */
in-port: OvenOn [deadline 2 sec], OvenOff [deadline 2 sec];
    /* Chicken */
var ToBeDone : time
begin
    start now within 20 min do
        call (RangeOn, nil);
        send (OvenOn, nil);
        receive (RangeOn, nil, 1);
        ToBeDone := now + 15 min ;
        delayedsend (OvenOff, ToBeDone, nil);
        from now to now+10min every 40 sec
            execute 10 sec within 10 sec do
                stir;
            end;
        call ( RangeOff, nil);
        receive(RangeOff, nil, 1);
        start after (ToBeDone-now) do end;
end;
end;

```

A large, friendly-looking blue cartoon robot character with a smiling face, two antennae-like appendages on its head, and a single long arm with a hand. It is standing on a light blue surface.

Code: Ada Implementation

A 3D wireframe model of a modern building with multiple levels and glass walls, rendered in white lines against a light blue background.

```

WITH Ada.Real_Time; USE Ada.Real_time;
WITH Ada.Calendar; USE Ada.Calendar;
PACKAGE CRS IS

```

```

    TASK Controller IS
        ENTRY get_order(prd_4_s,: in second;
                        prd_4_b: in time;
                        prd_4_f: in time);
        ENTRY serve_with_food();
    END Controller;

```

```
END CRS;
```

```
PACKAGE BODY CRS IS
```

```
-- ...
BEGIN
```

```

    TASK BODY Controller IS
        s_period, b_period, f_period: time := 0;
        end_t, current_t: Clock:= 0;
    BEGIN
        LOOP
            ACCEPT get_order(prd_4_s: in time;
                            prd_4_b: in time;
                            prd_4_f: in time) DO
                s_period := prd_4_s;
                b_period := prd_4_b;
                f_period := prd_4_f;
            end;
            current_t := clock();
            end_t := current + period_4_s;
            Oven.bake(b_period);
            Range.fry(f_period);
            ACCEPT fry_done();
            ACCEPT bake_done();
            SELECT
                DELAY end_t;
            OR
                ACCEPT serve_with_food();
            END SELECT;
        END LOOP;
    END Controller;

```

```

    TASK BODY Oven is
        start_t, end_t : Clock;
        period : time;
    BEGIN
        LOOP
            ACCEPT bake(prd: in time) DO
                period := prd;
            END;
            start_t := Clock();
            end_t := start_t + period;
            SELECT
                DELAY end_t;
                -- Baking chicken
            OR
                Controller.bake_done();
            END SELECT;
        END LOOP;
    END Controller;

```

```

    TASK Range IS
        ENTRY fry(prd: in time);
    END Controller;

```

```

    TASK BODY Range is
        current_t, start_t, end_t : Clock;
        period, next : time;
    BEGIN
        LOOP
            ACCEPT fry(prd: in time) DO
                period := prd;
            END;
            start_t := Clock();
            end_t := start_t + period;
            -- Frying potatoes
        LOOP
            current := clock();
            if (end_t > current + 40)
                -- Stirring for 40 Seconds
            else
                EXIT done;
            END LOOP;
        done: SELECT
            DELAY end_t;
        OR
            Controller.fry_done();
        END SELECT;
    END LOOP;
    END Controller;

```

```
END CRS.
```

2005-09-15