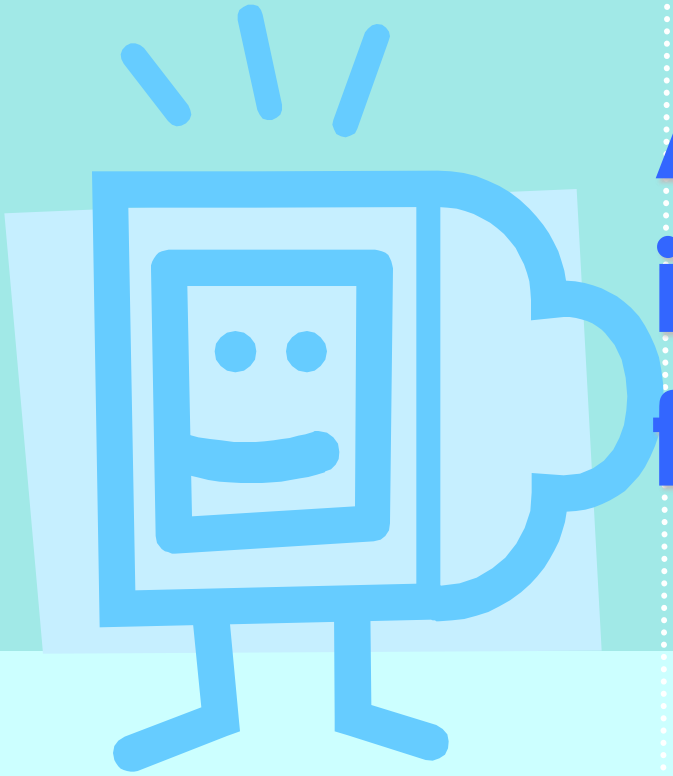
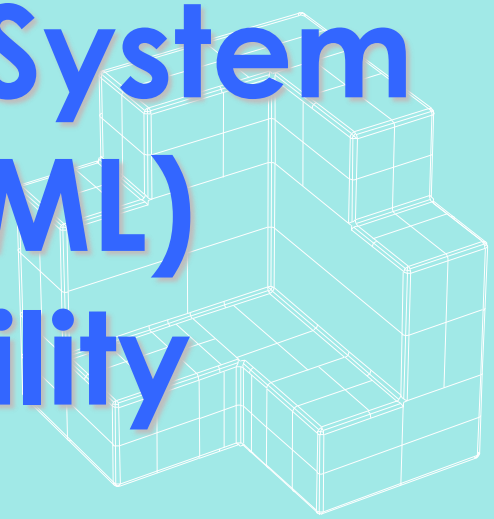


A Cooking System in ROPES (UML) for Traceability



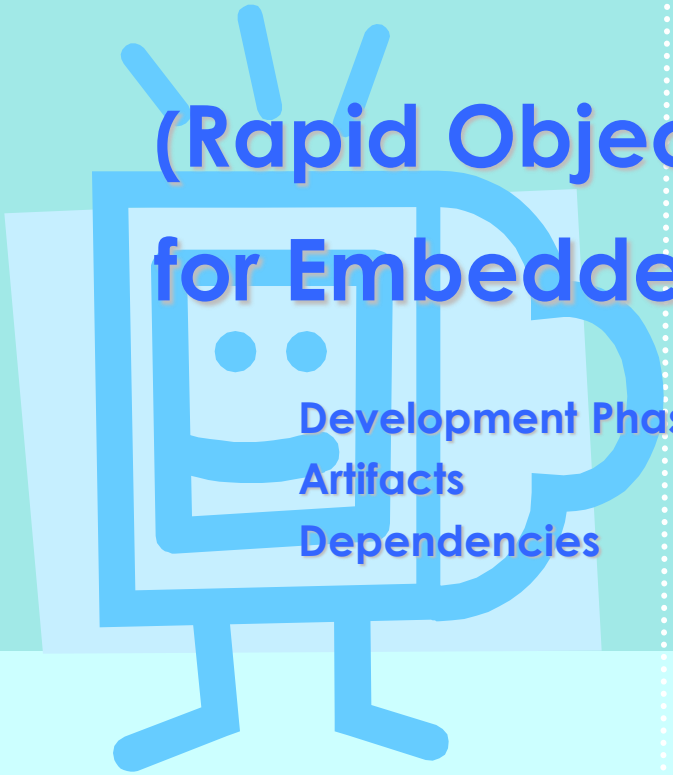
Moon Kun Lee
Computer Engineering
Chonbuk National University

Outline

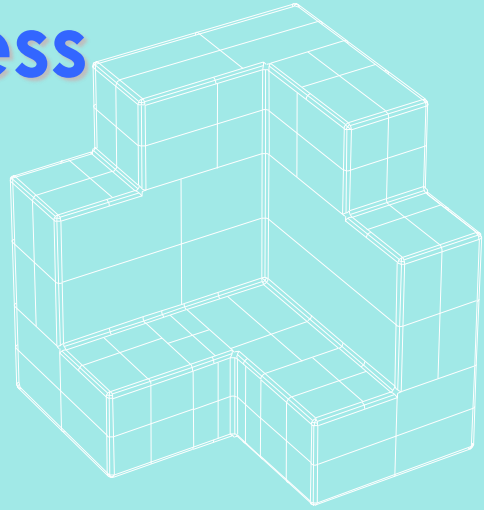
- ROPES
- Cooking System Example

ROPES

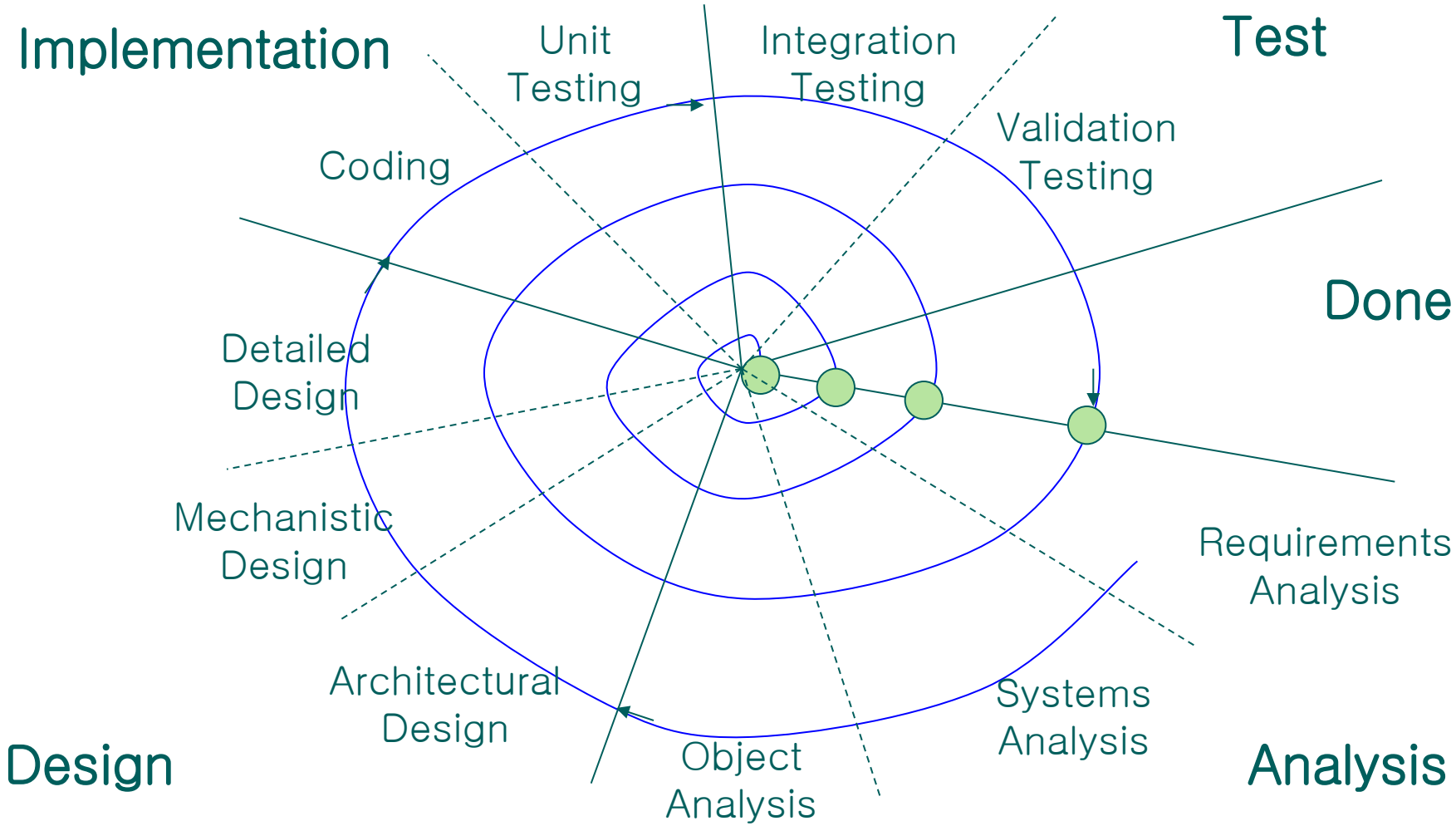
(Rapid Object-Oriented Process for Embedded Systems)



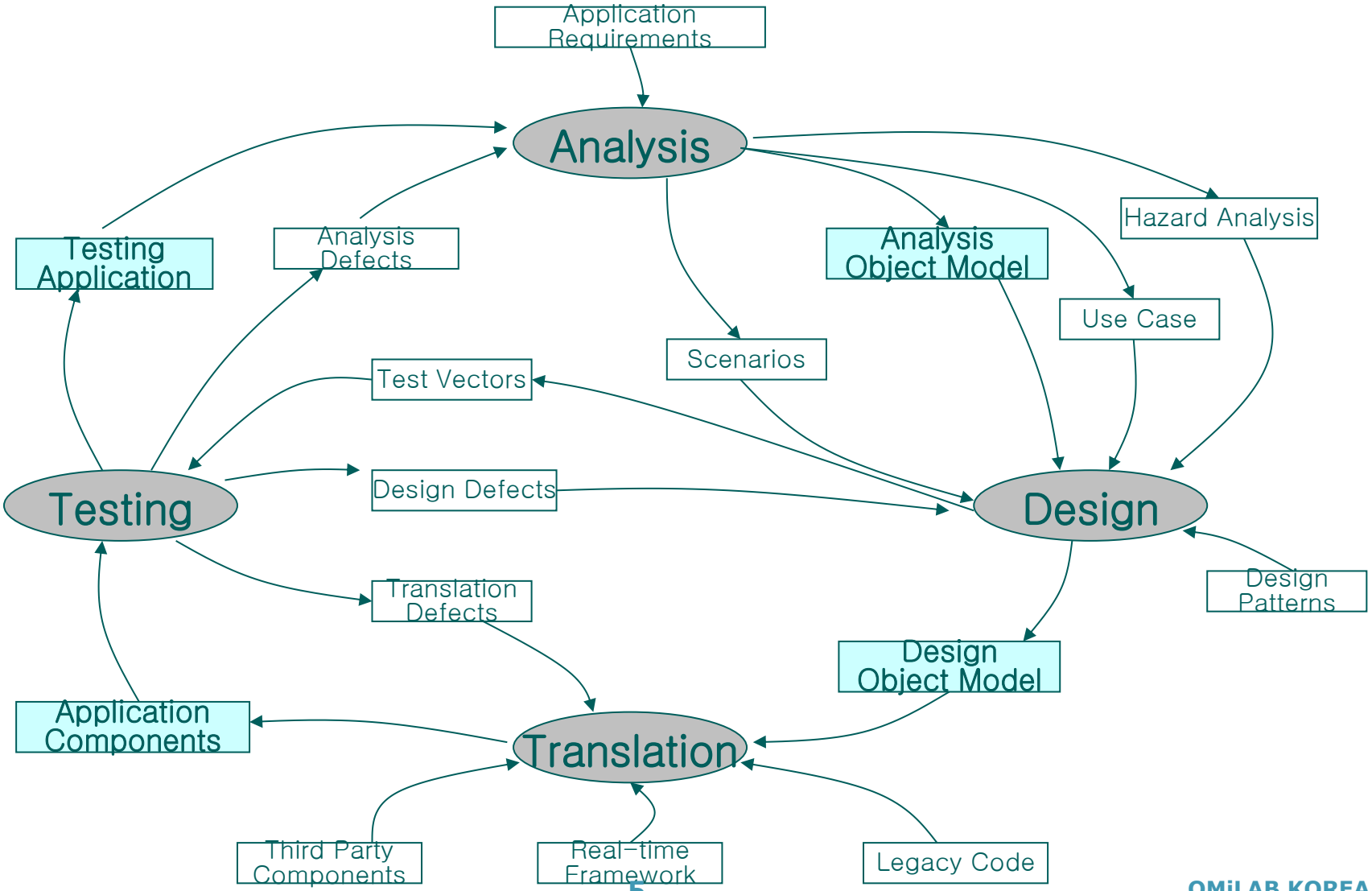
Development Phase
Artifacts
Dependencies



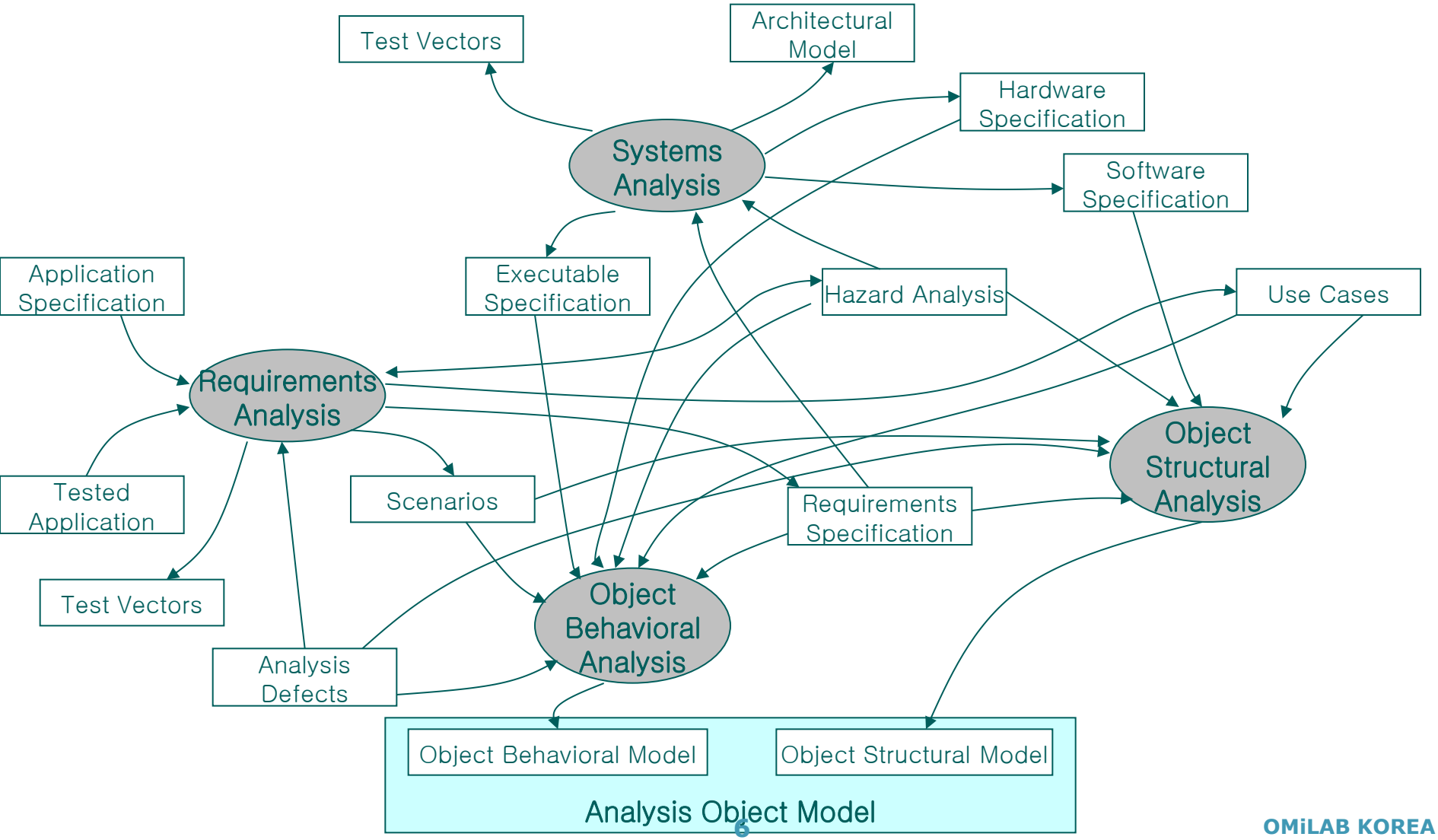
ROPES Process



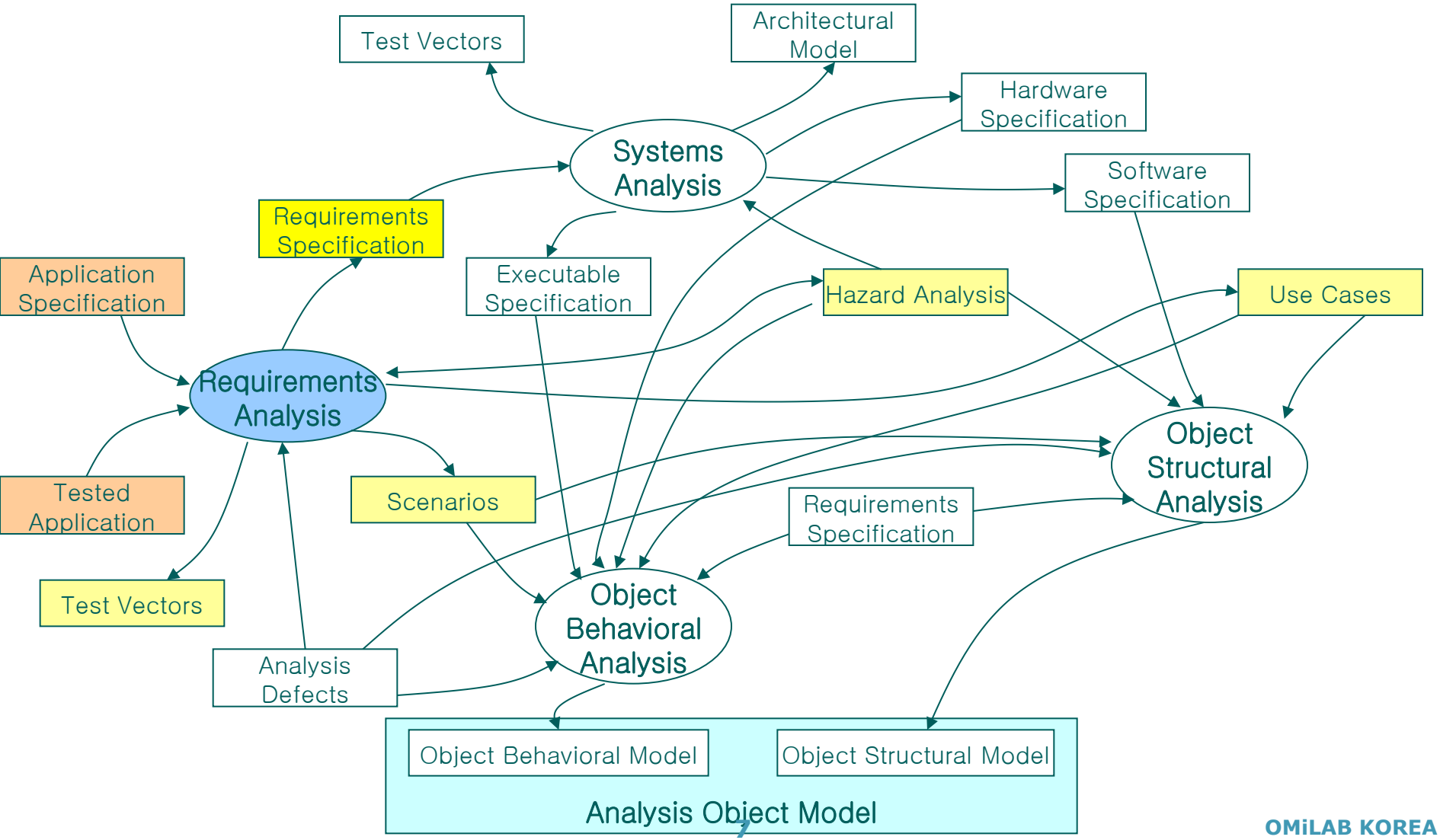
ROPES Process Artifacts



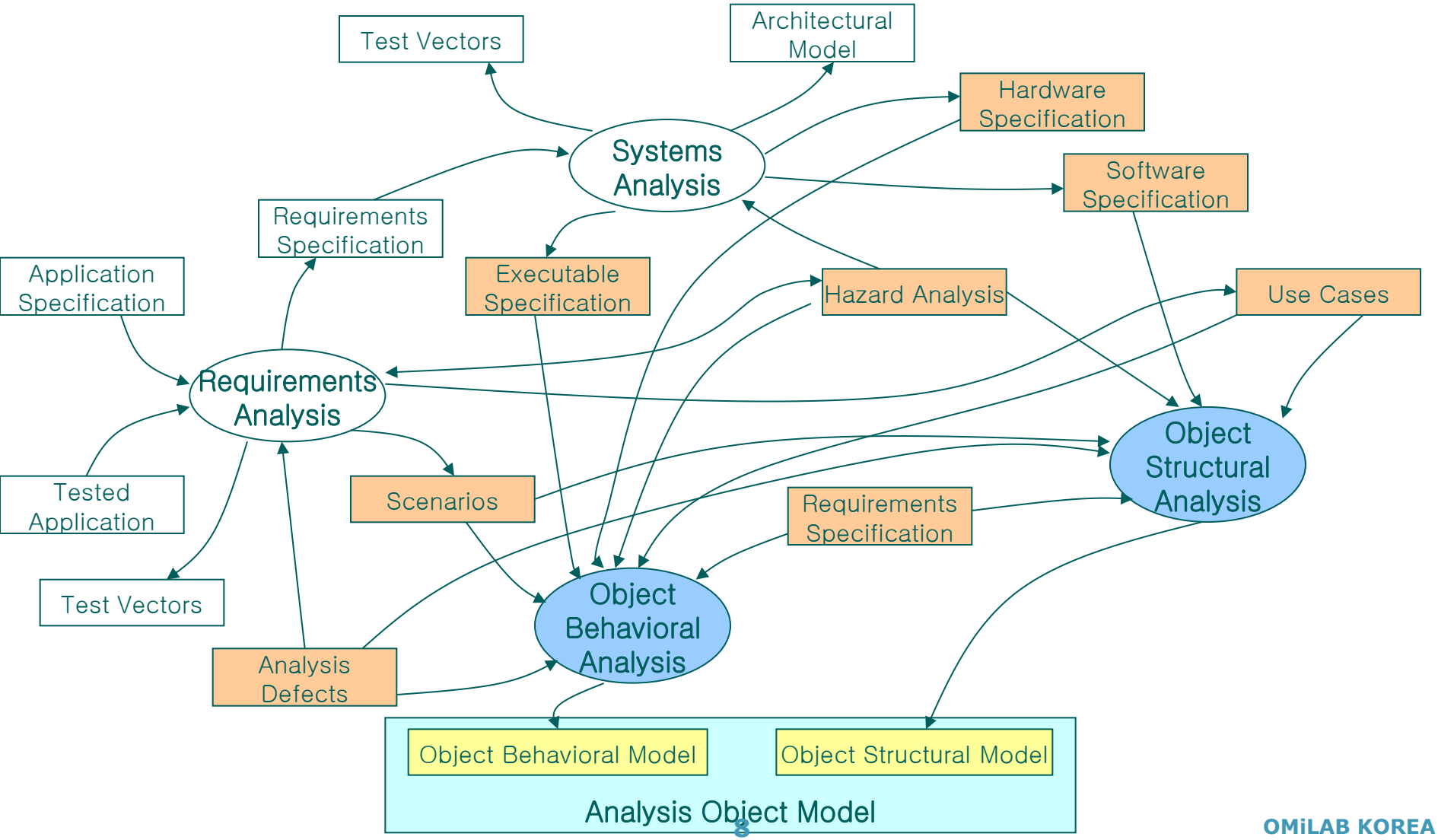
ROPES Analysis Model Artifacts



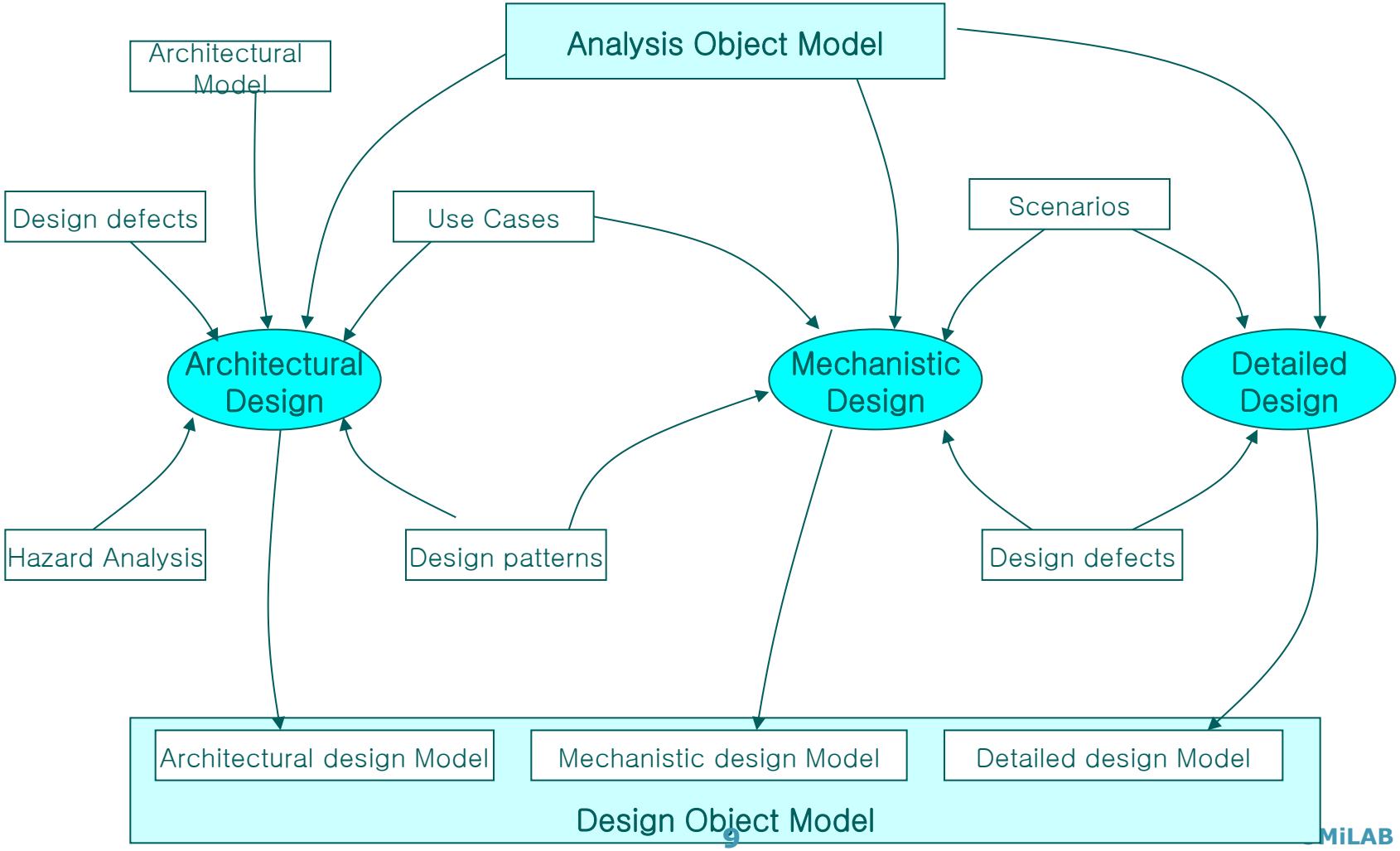
Requirements Analysis



Object Structural/Behavioral Analysis

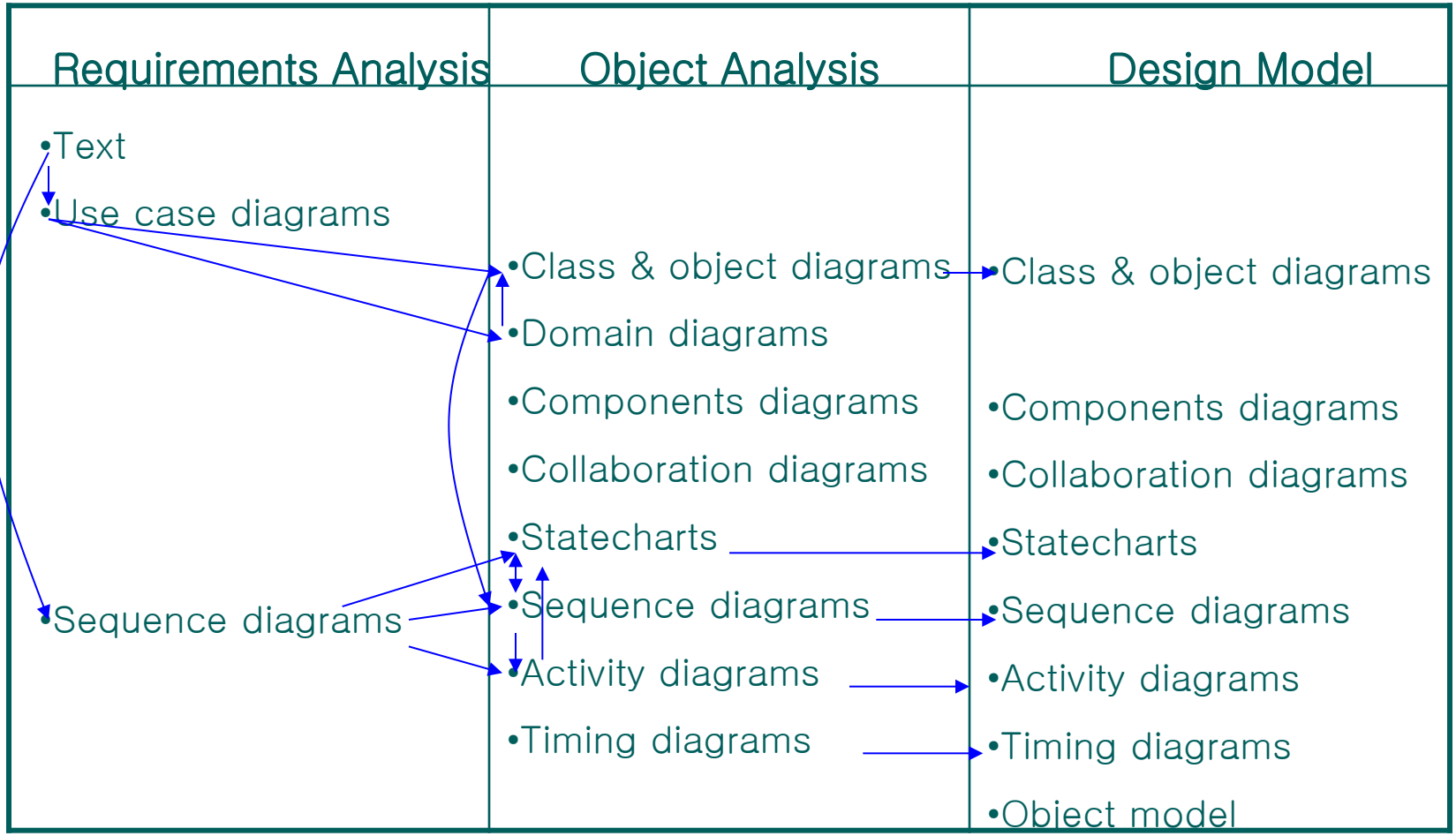


ROPES Design Model Artifacts



MiLAB KOREA

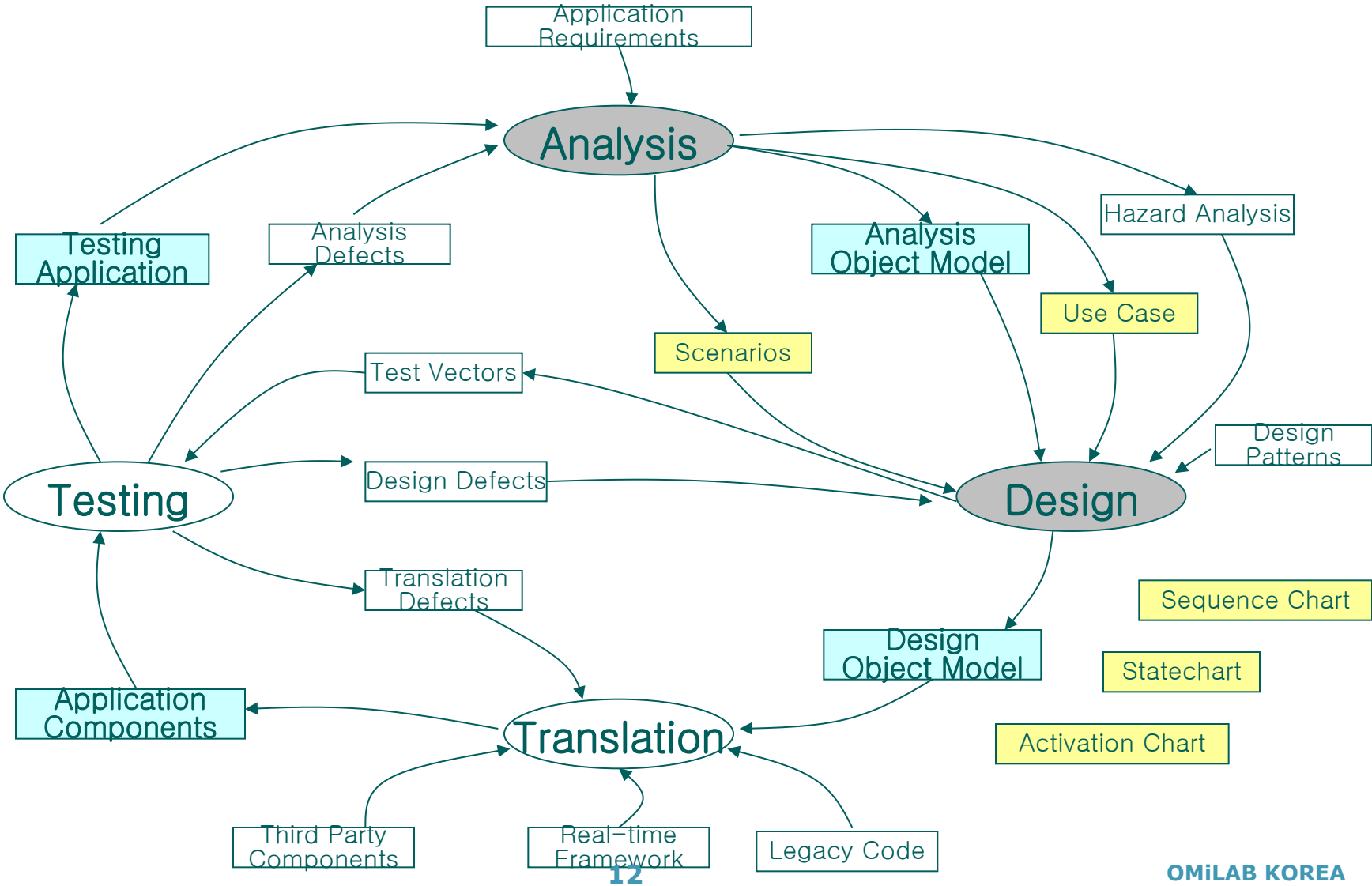
Artifacts among Phases



Cooking Example in ROPES

Development Phases
Artifacts
Dependencies

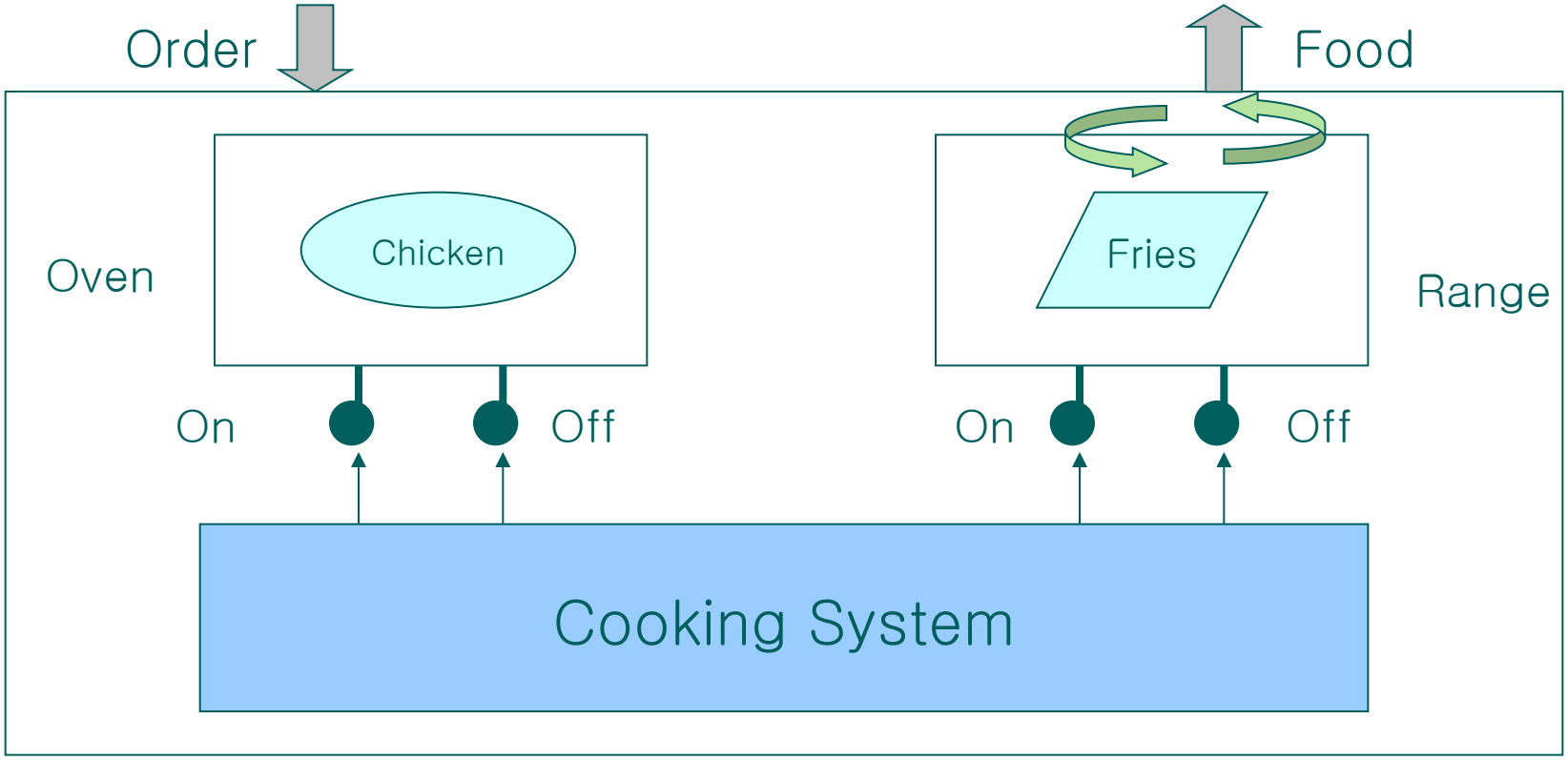
Process View

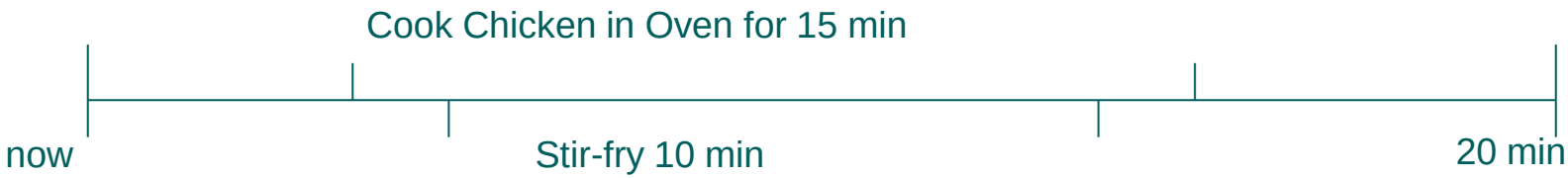


Requirements

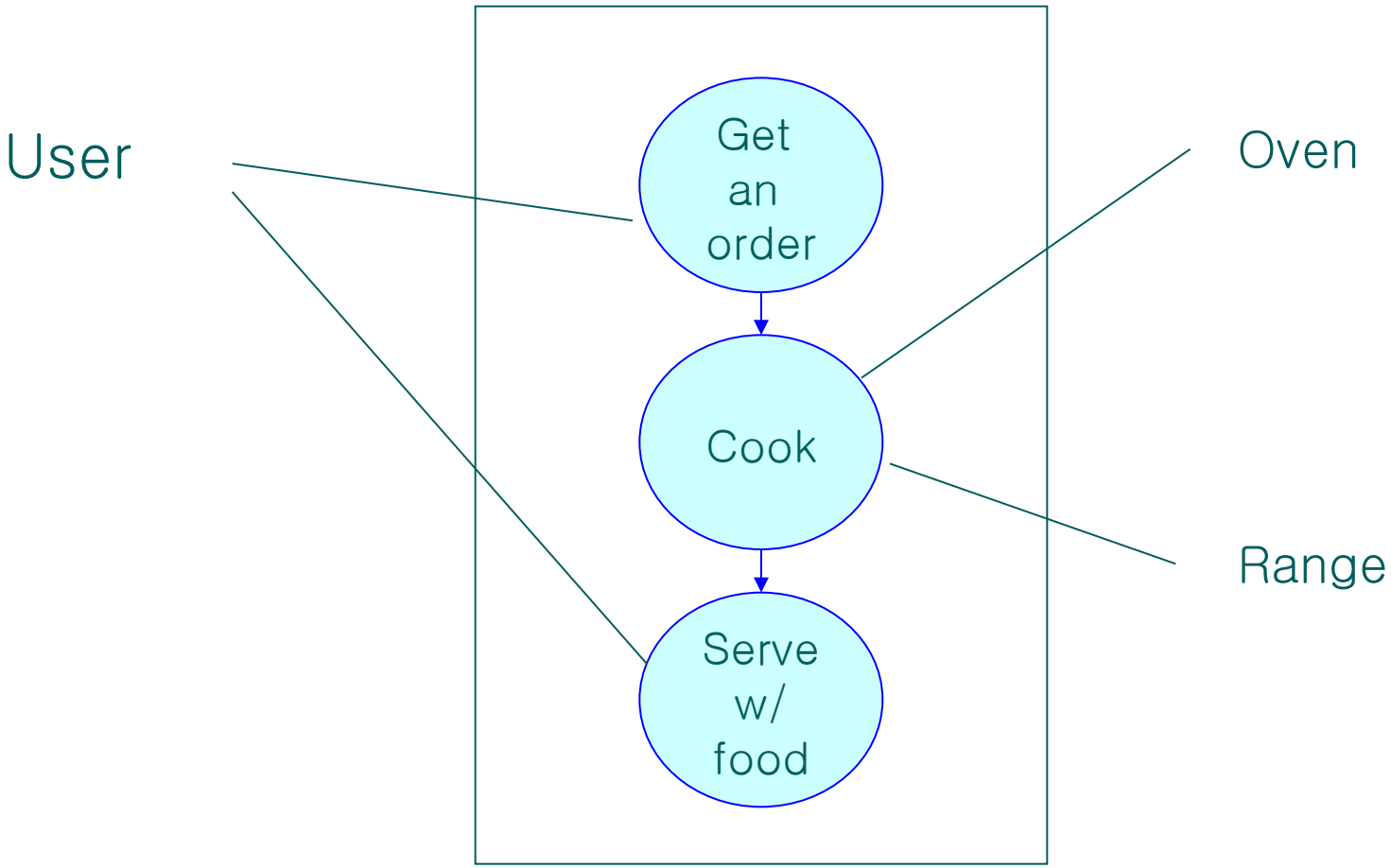
1. The following is a list of the requirements for a cooking system.
2. When a customer orders, the system provides him/her with a piece of chicken and a bag of French fries in 20 minutes.
3. Cooking the chicken takes 15 minutes in an oven.
4. Frying takes 10 minutes in a range.
5. Cooking and frying 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

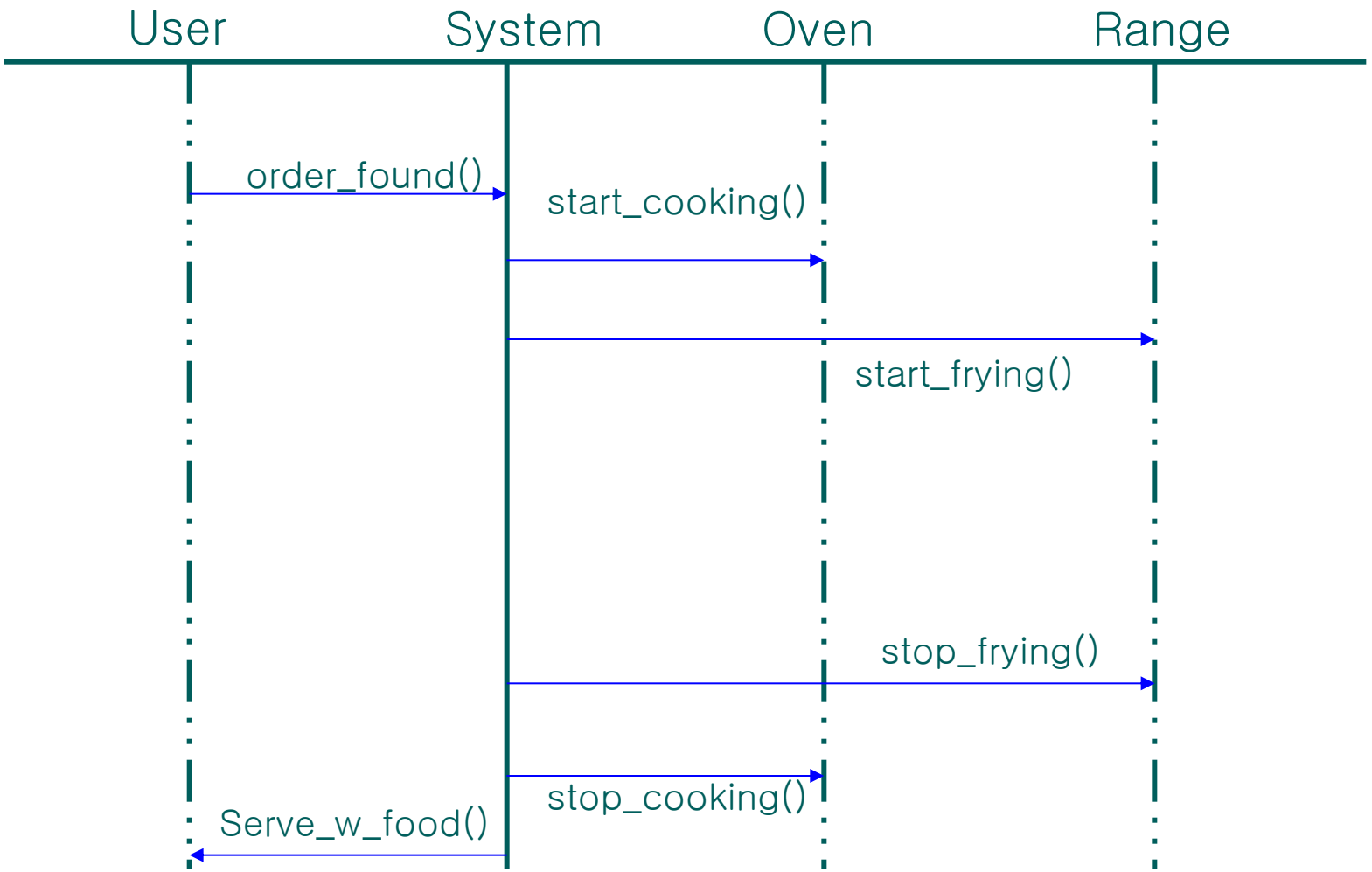

Order
Food
Oven
Chicken
Fries
Range
On
Off
On
Off
Cooking System


Cook Chicken in Oven for 15 min
now
Stir-fry 10 min
20 min

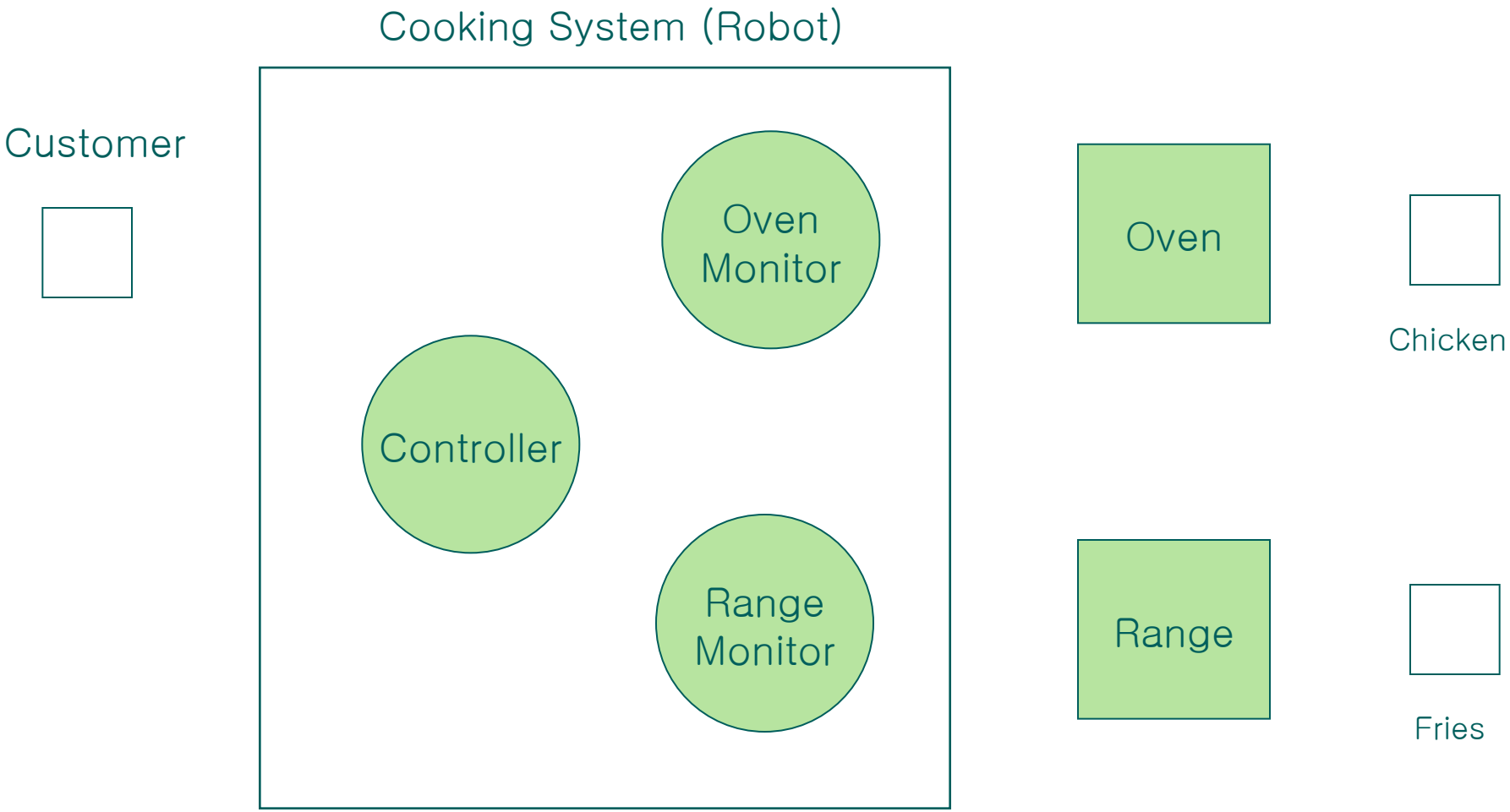
Use-Case



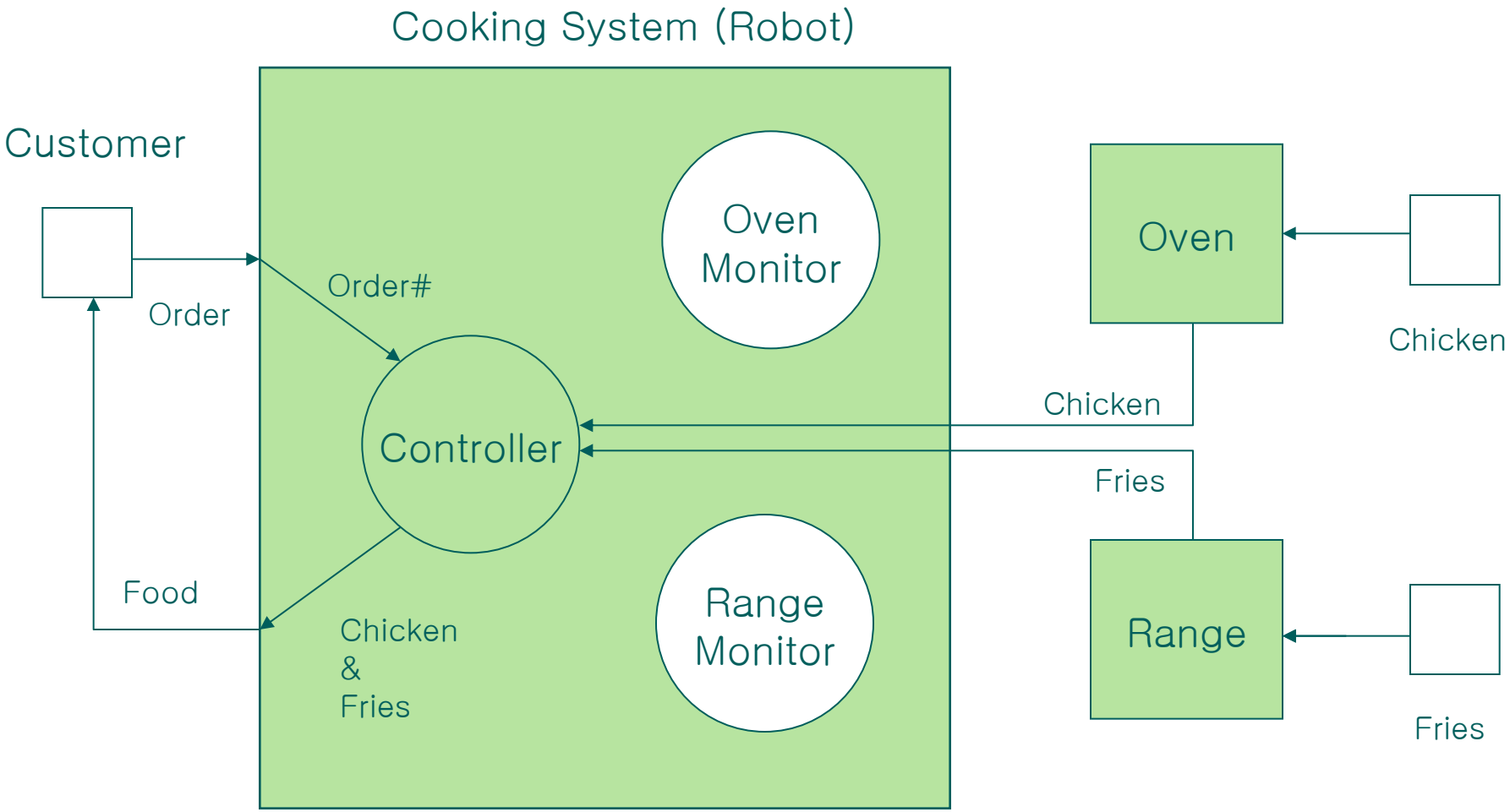
Scenario


User
System
Oven
Range
order_found()
start_cooking()
start_frying()
stop_frying()
stop_cooking()
Serve_w_food()

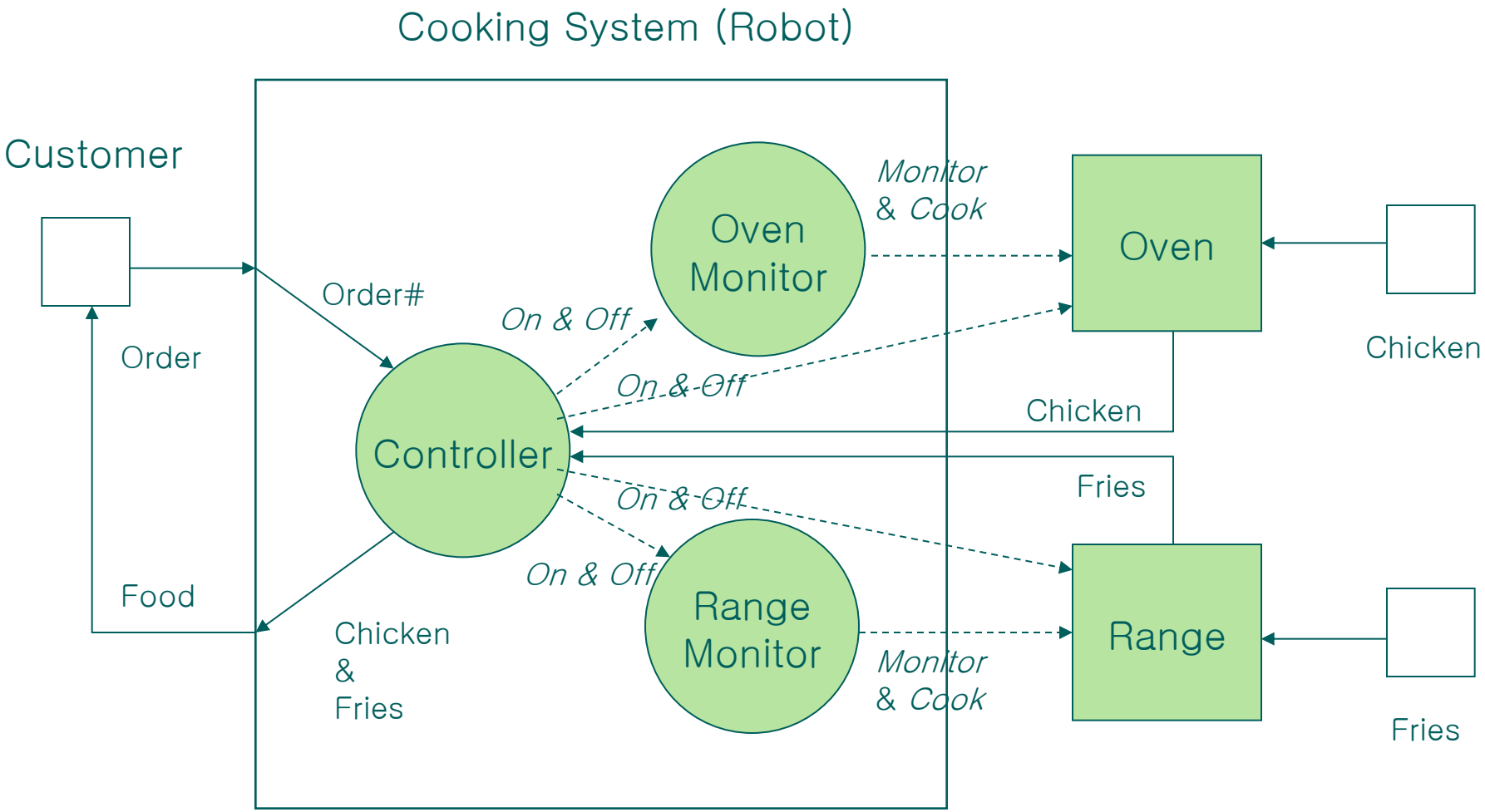
Structural


Cooking System (Robot)
Customer
Oven Monitor
Controller
Range Monitor
Oven
Chicken
Range
Fries

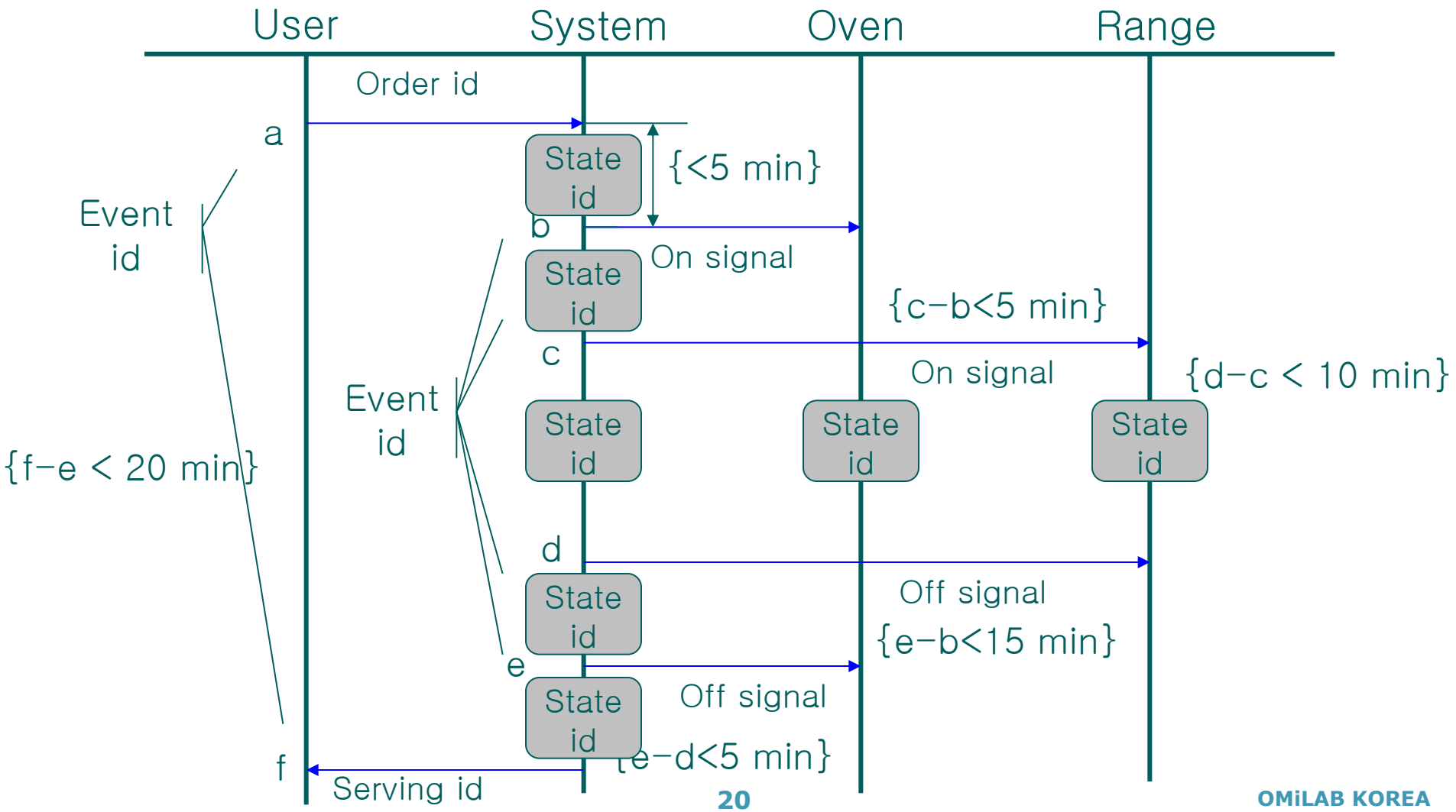
Functional


Cooking System (Robot)
Customer
Order
Order#
Oven Monitor
Controller
Range Monitor
Food
Chicken & Fries
Oven
Chicken
Chicken
Fries
Range
Fries

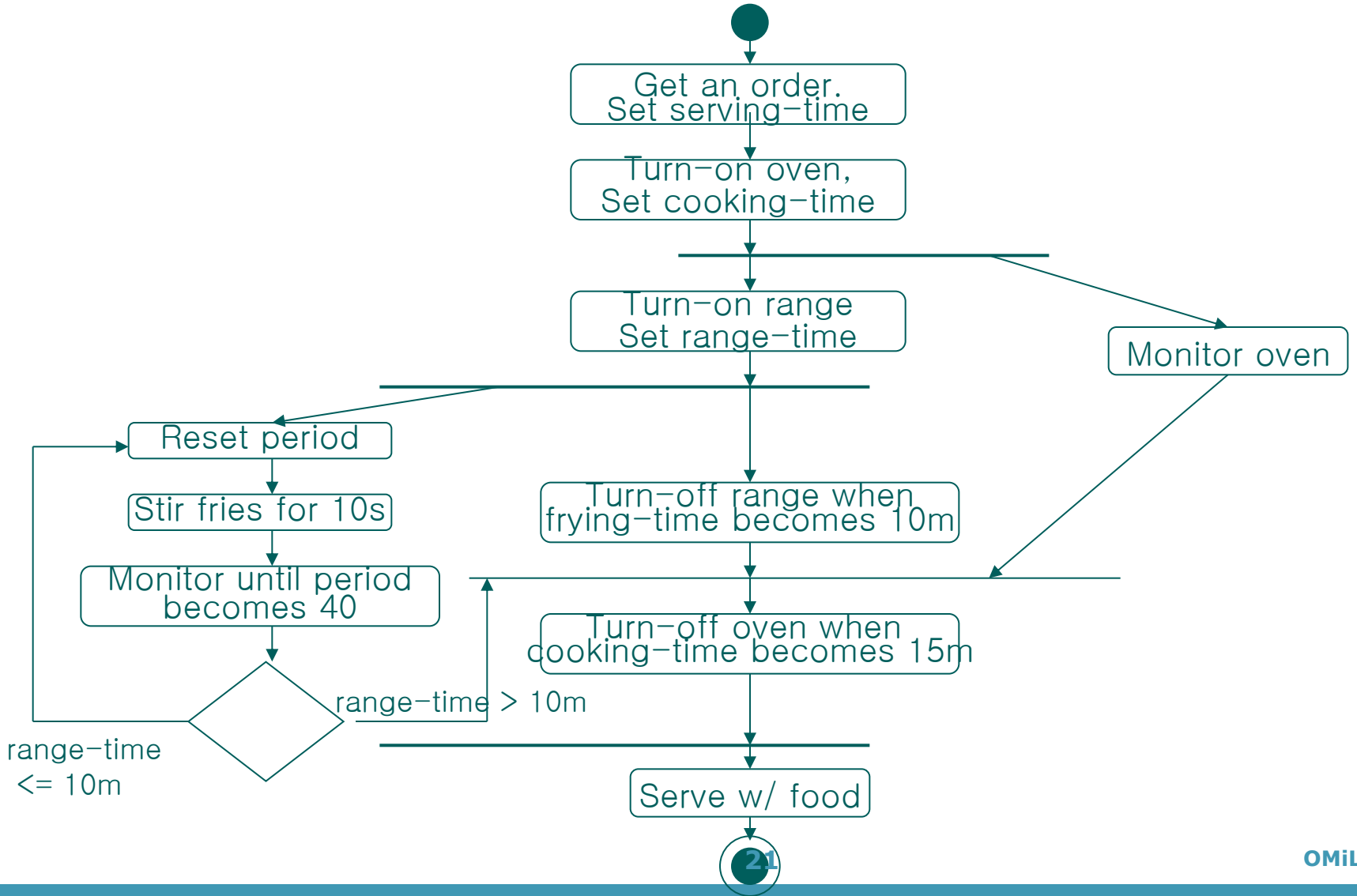
Control Flow Diagram



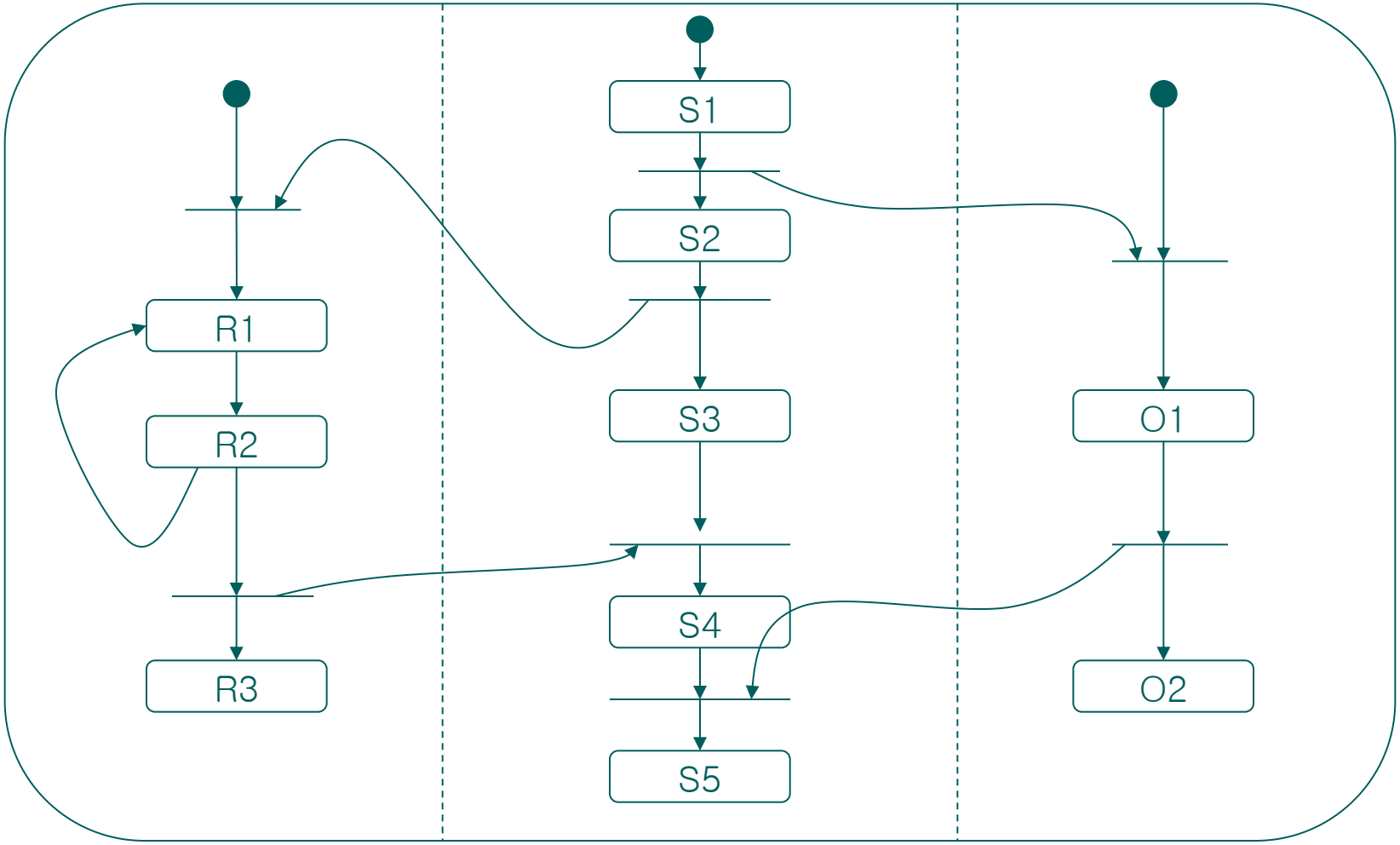
Sequence Chart



Activity Chart



Statechart


S1
S2
S3
S4
S5
R1
R2
R3
O1
O2
Range monitor
Controller
Oven monitor

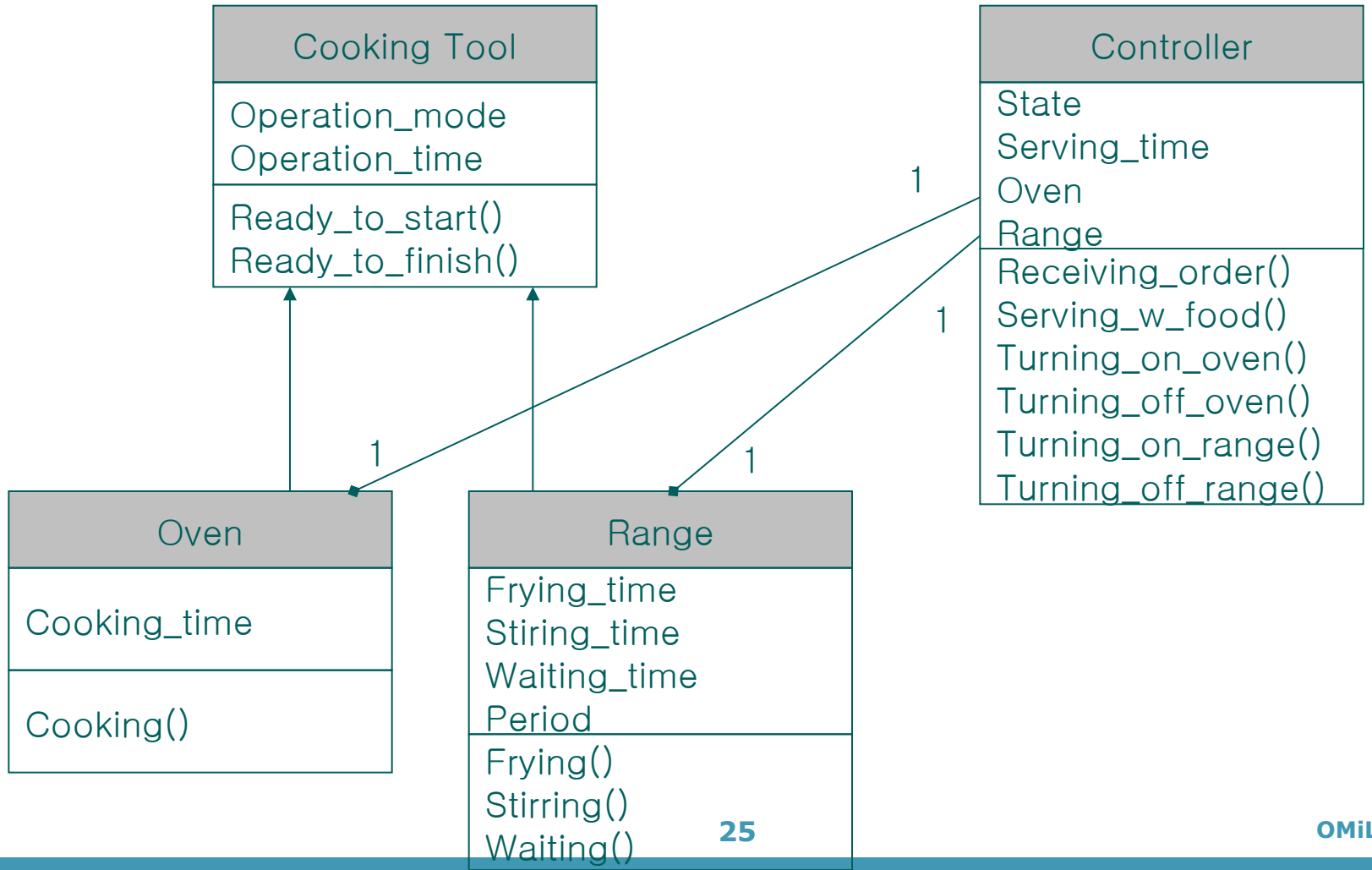
Object Identification

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(cooking_time)=15m c(fryng_time)=10m	Cooking() Frying() Stirring()	
Oven	Operation_mode Operation_time Cooking_time	c(cooking_time)=15m	Cooking()	
Range	Operation_mode Operation_time Frying_time Stirring_time Period Waiting_time	c(fryng_time)=10m c(Stirring_time)=10s c(Period)=40s	Frying() Stirring() Waiting()	

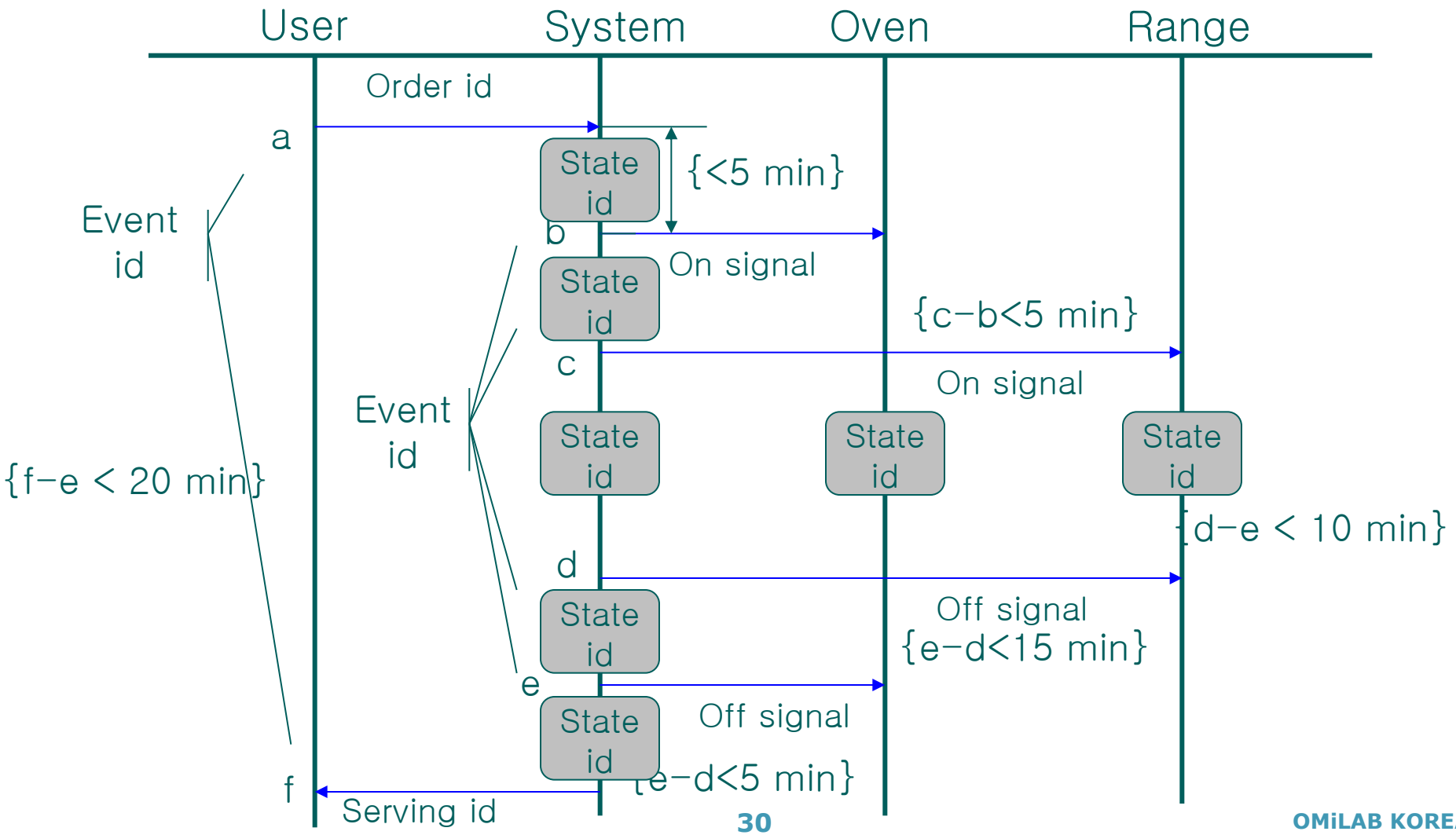
Timing/Ordering Requirements

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. R3−1
4. **Frying** takes 10 minutes in a range. R4−1
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**. R7−1 R7−2

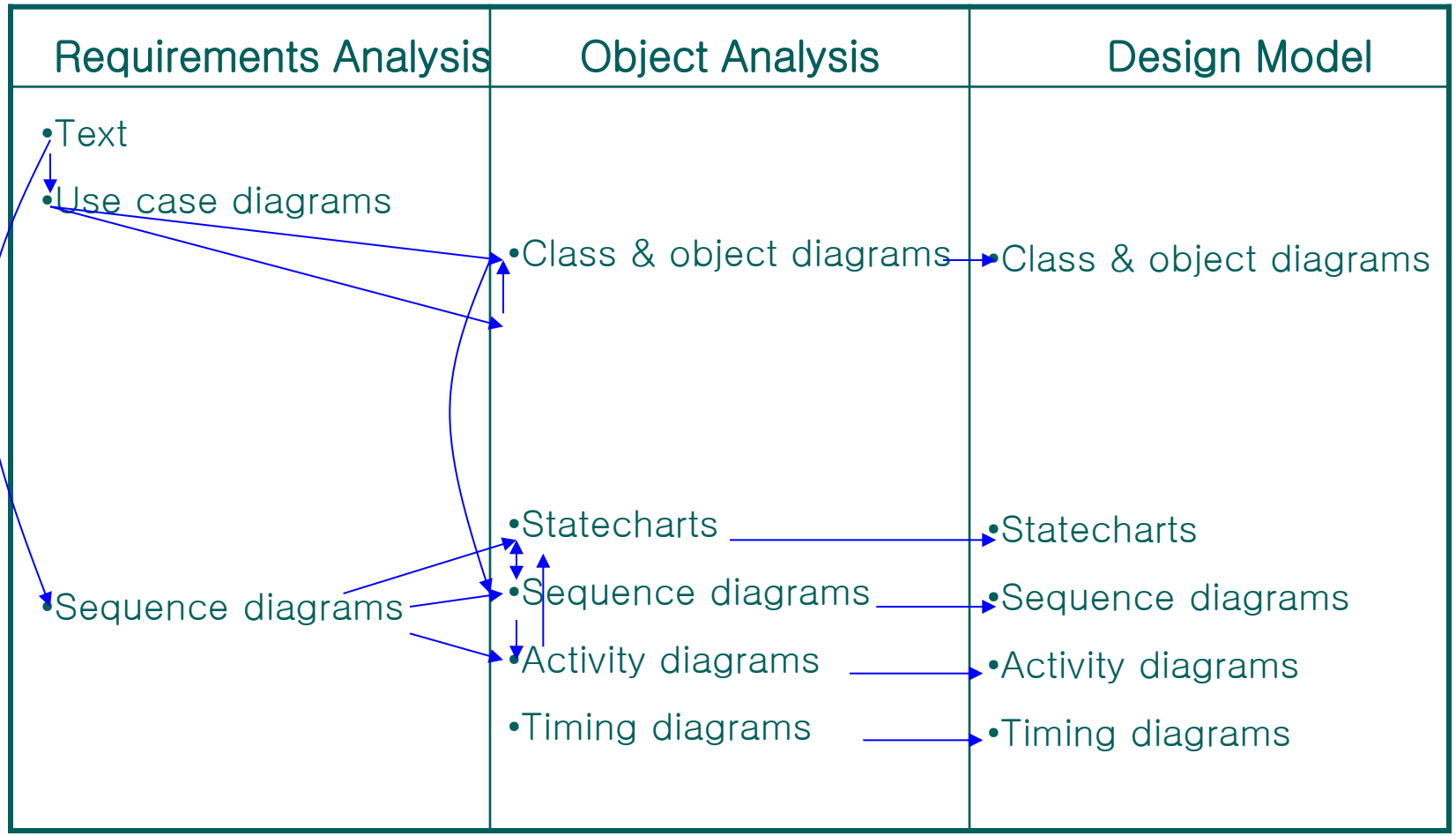
Relation Requirements Properties

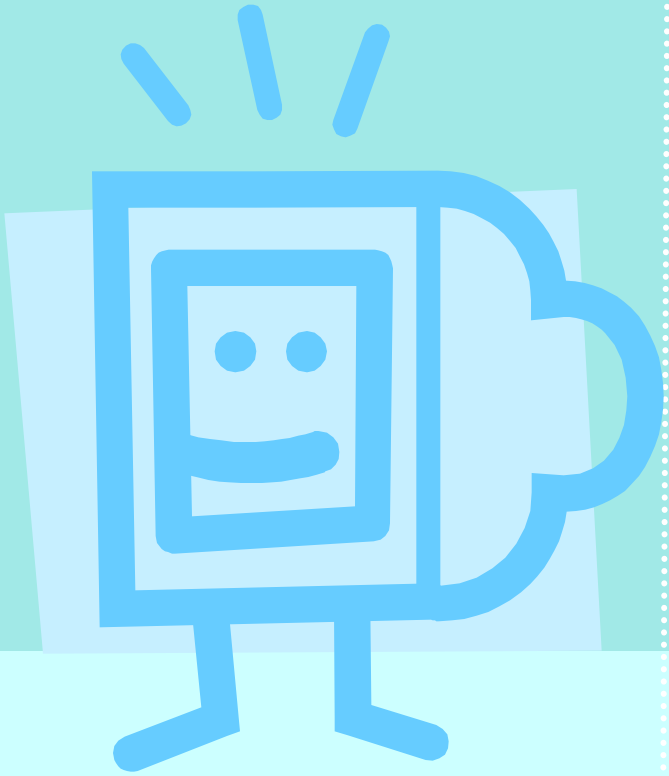
Req	Properties
R2-1	t(controller'order) - t(controller'serve) < 20m
R3-1	t(oven'operation == off) - t(oven'operation == on) < 15m
R4-1	t(range'operation == off) - t(range'operation == on) < 10m
R7-1	t(oven'operation == on) - t(range'operation == on) >= 0s
R7-2	t(oven'operation == off) - t(range'operation == off) <= 0s

Sequence Diagram

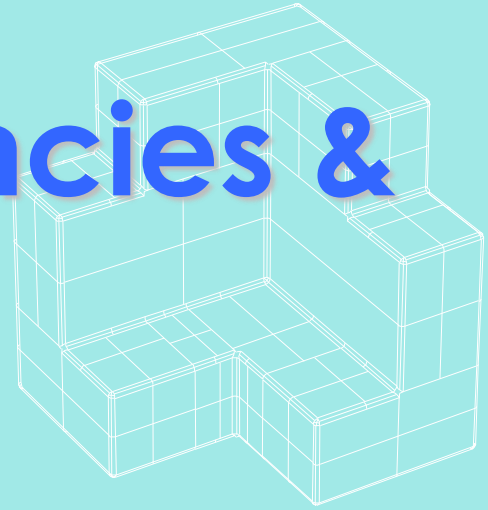


Relations among Artifacts





Dependencies & Relations



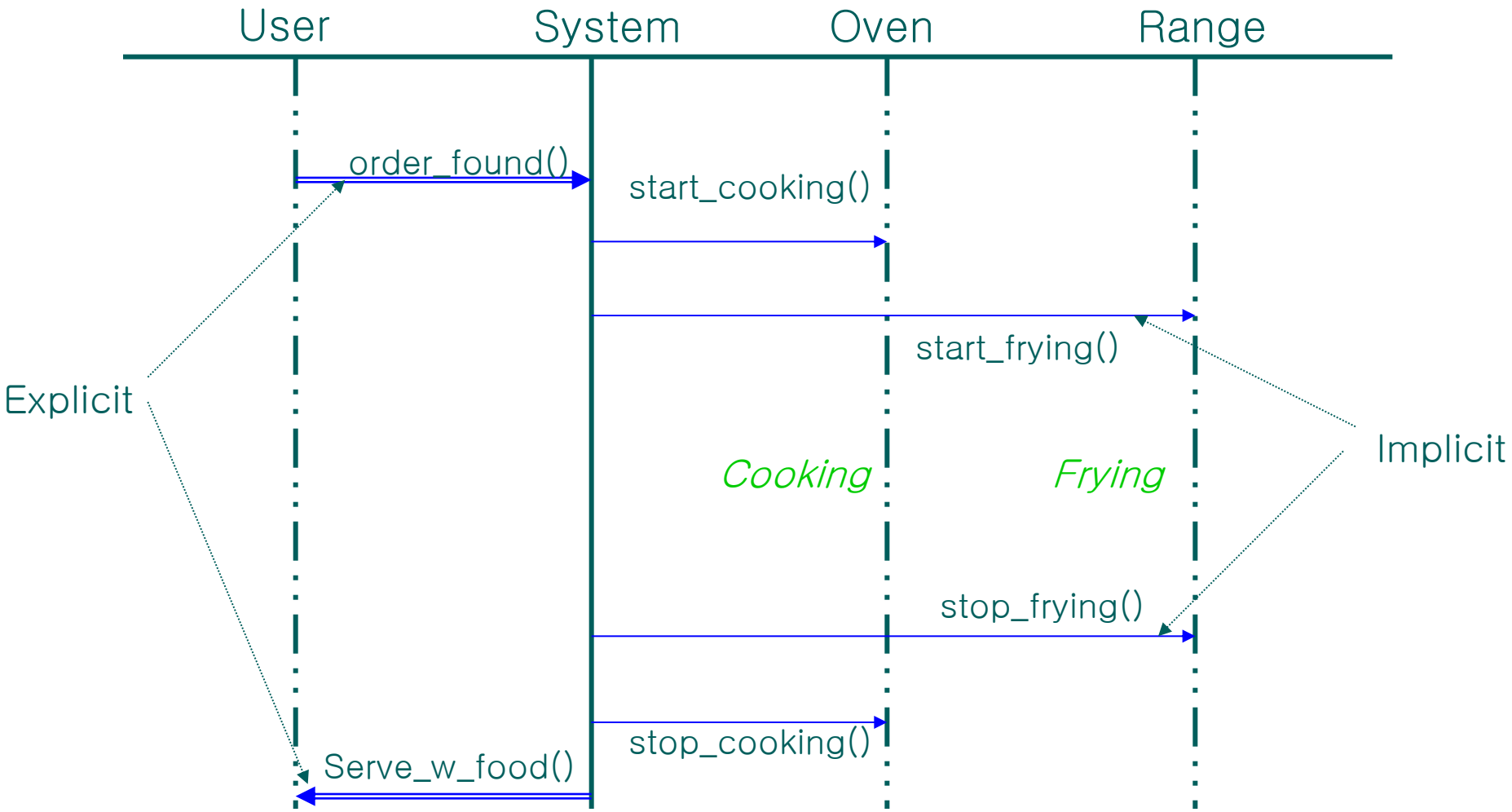
Dependency

- System components
- Interaction/event
- Ordering/timing
- Explicit/inplicit
- 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.

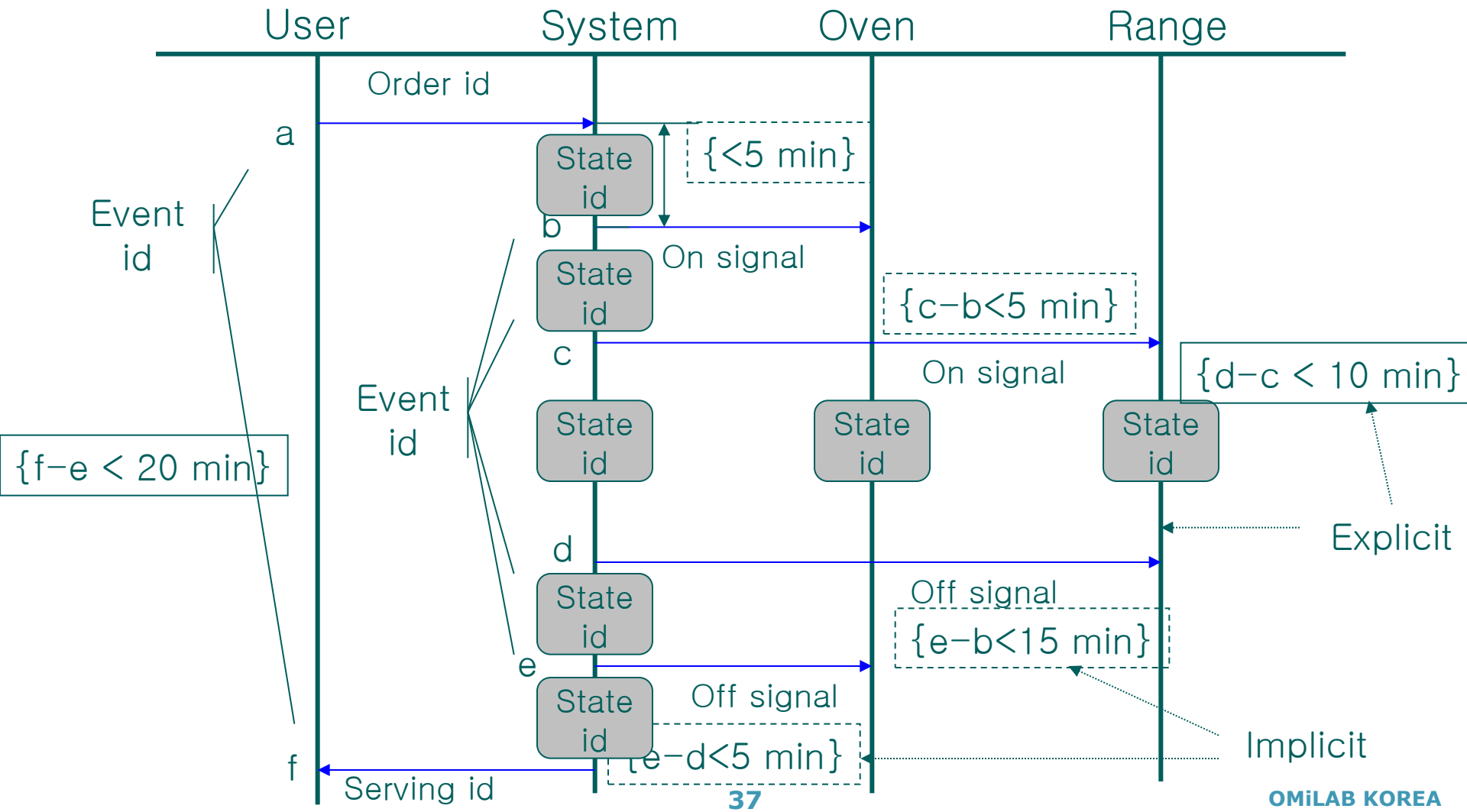
Scenario


User
System
Oven
Range
order_found()
start_cooking()
start_frying()
Explicit
Implicit
Cooking
Frying
stop_frying()
stop_cooking()
Serve_w_food()

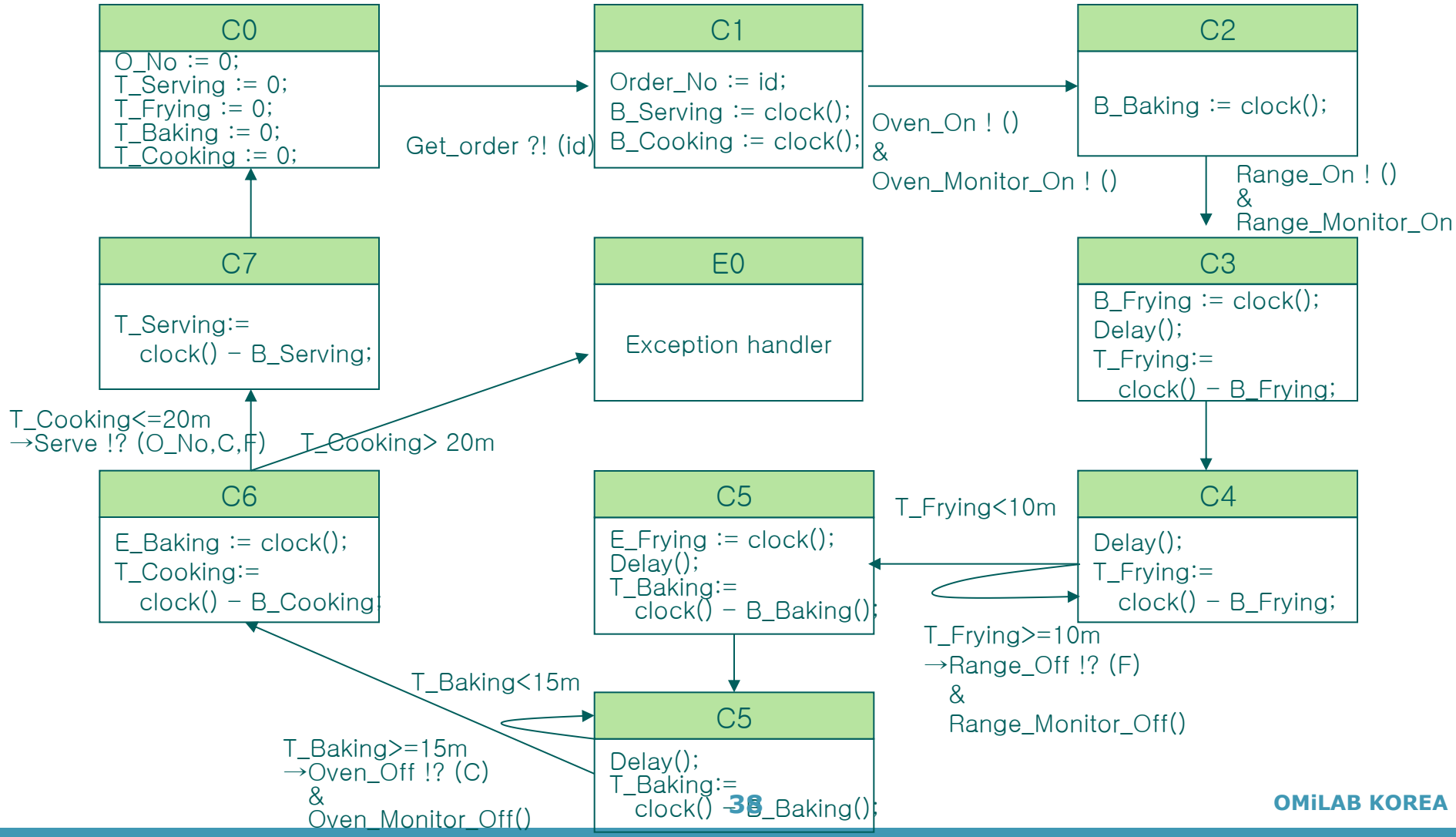
Ordering/Timing

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

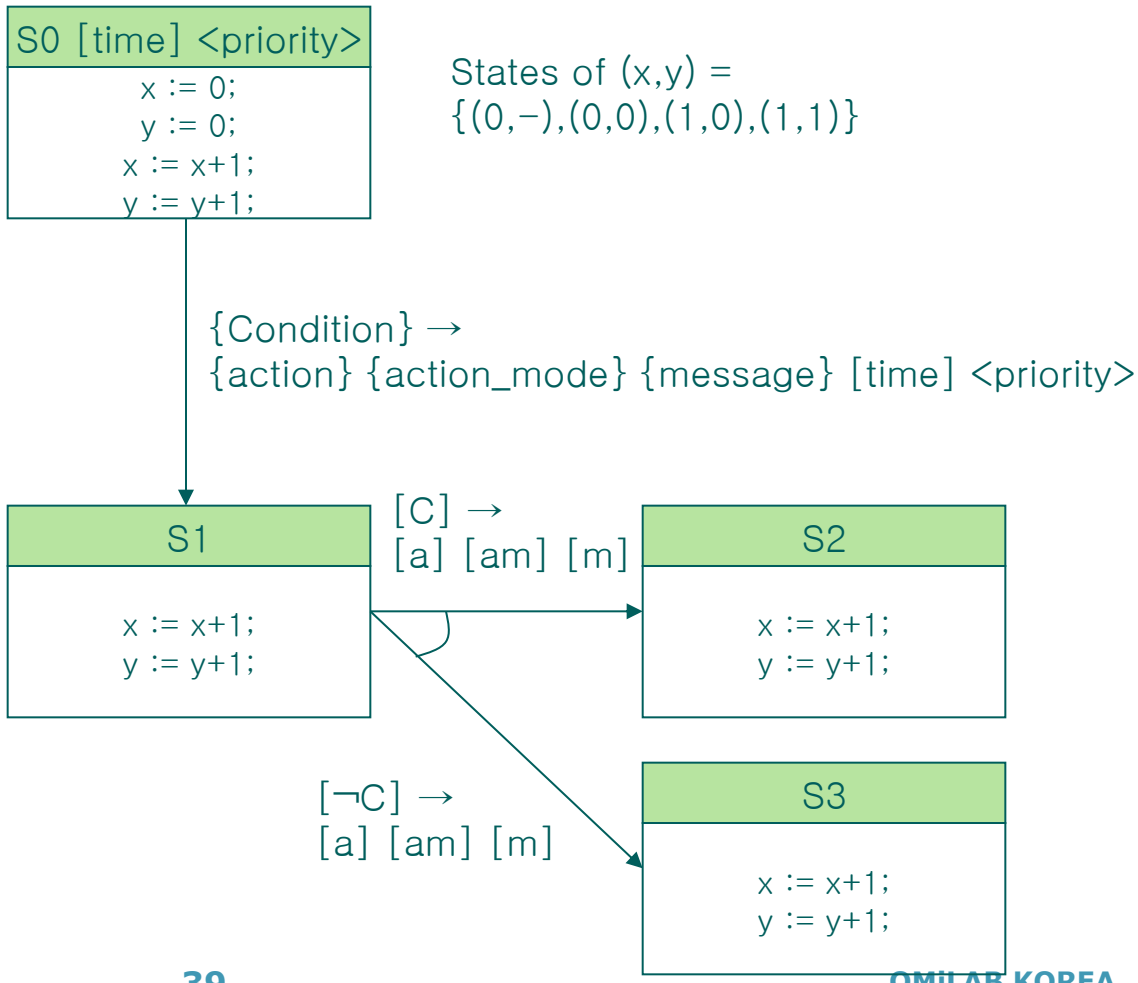


State Machine: Controller

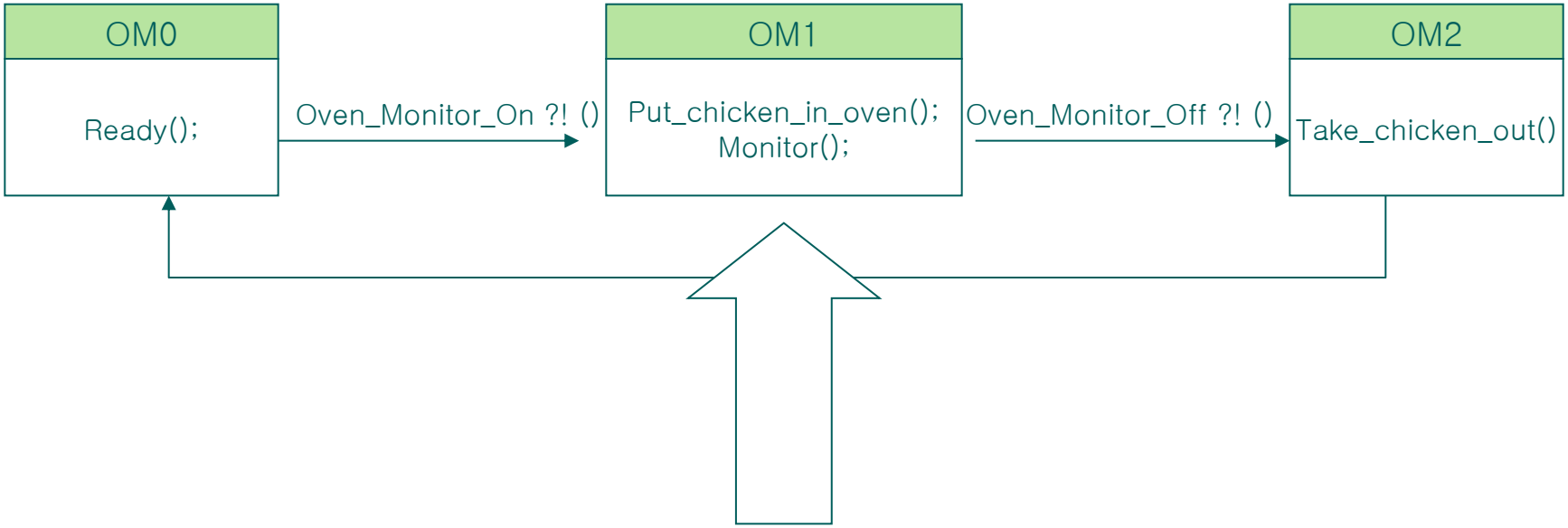


SM: Introduction

- Space:
  - set of states
  - properties:
    - + time
    - + priority
- Transition:
  - change of space
  - properties:
    - + time
    - + priority
- Action:
  - mode:
    - + asynchronous:
      - send: !
      - receive: ?
    - + synchronous:
      - send: !?
      - receive: ?!

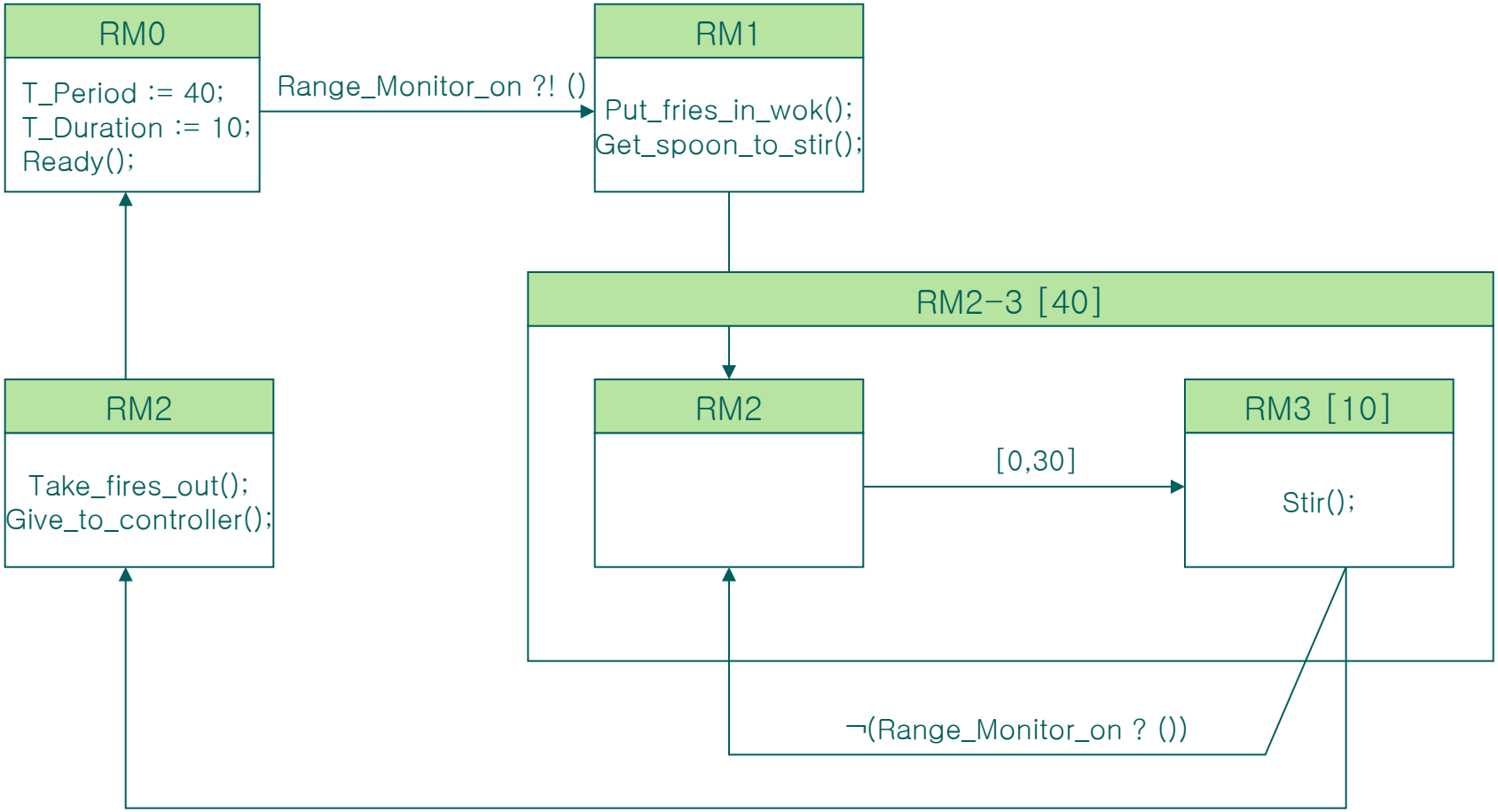


Oven Monitor



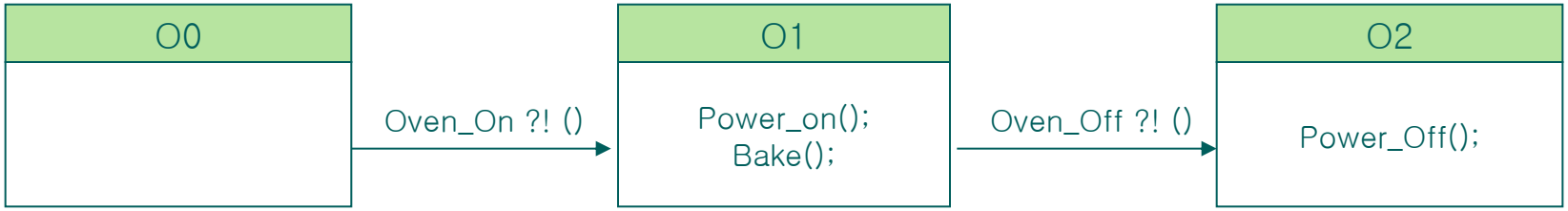
Monitoring activies can be specified,
i.e., temperature, faces, distance, etc.

Range Monitor

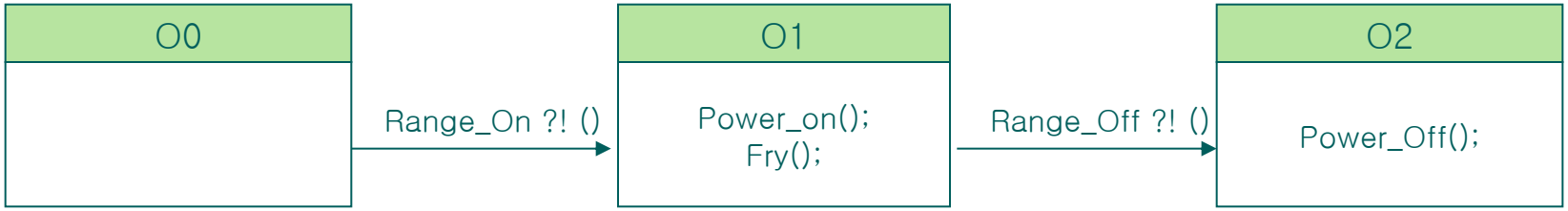


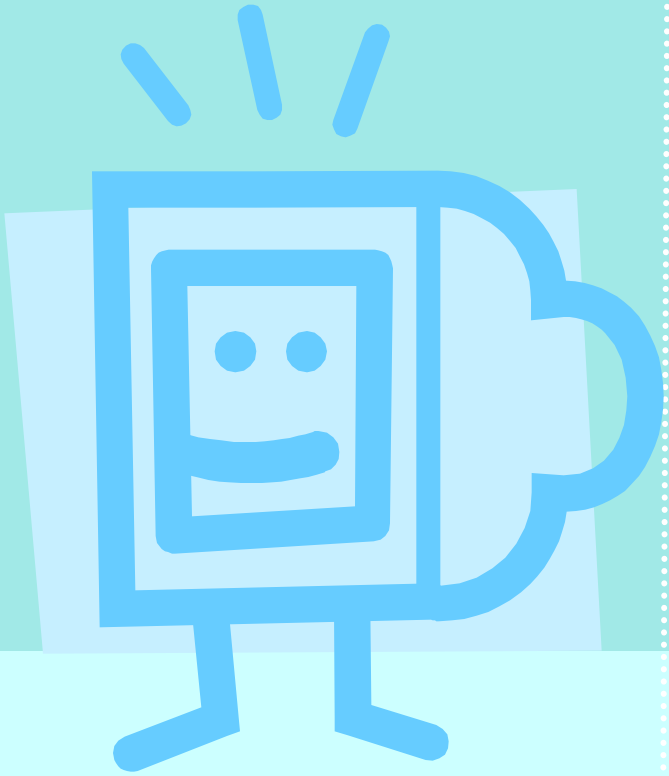
Range_Monitor_on ? ()

Oven

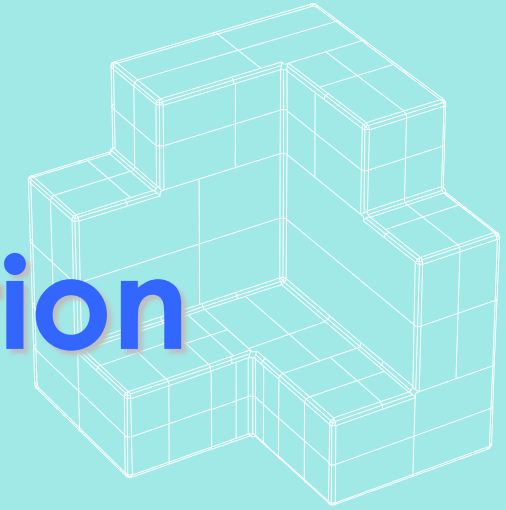


Range



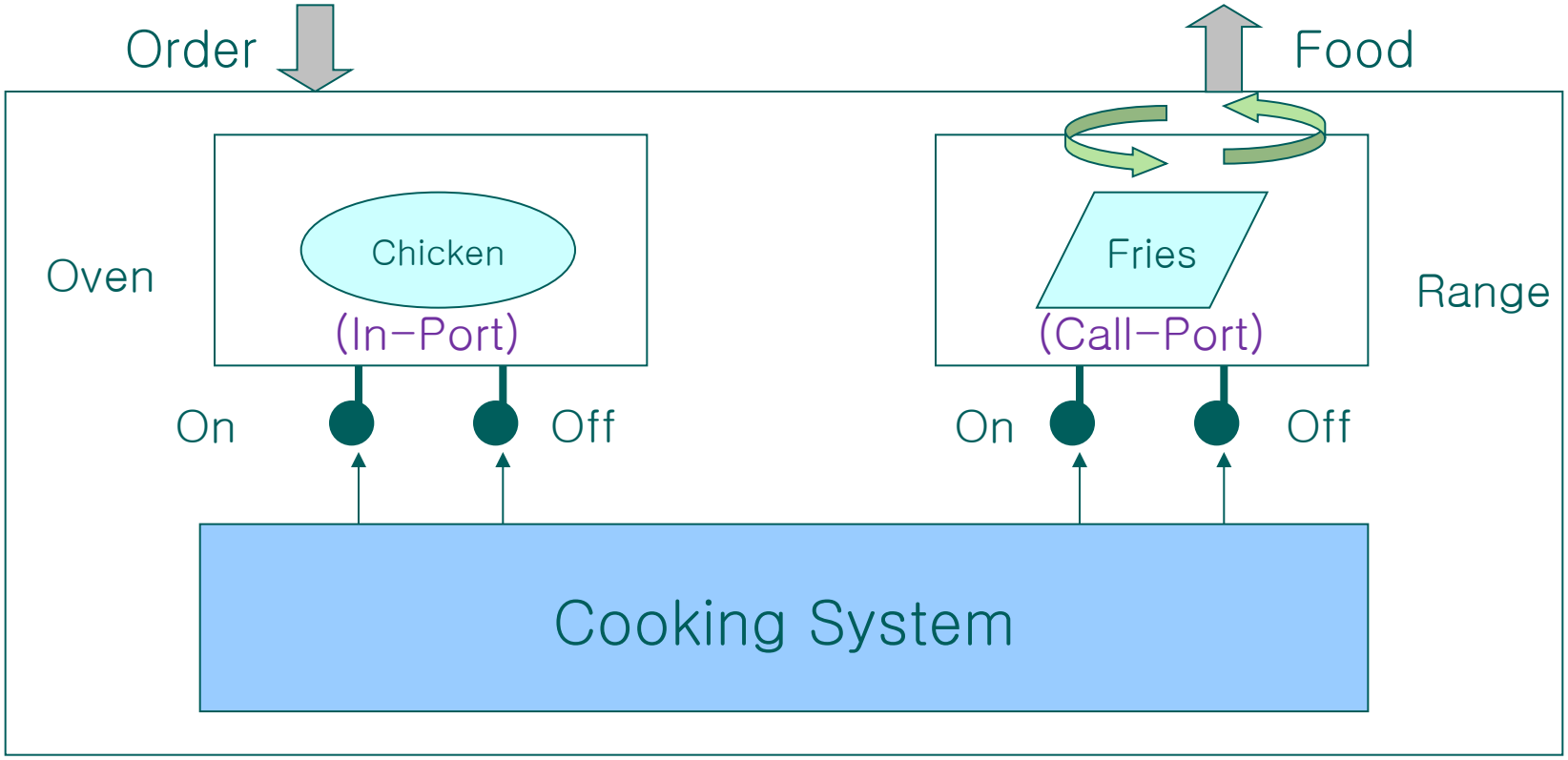


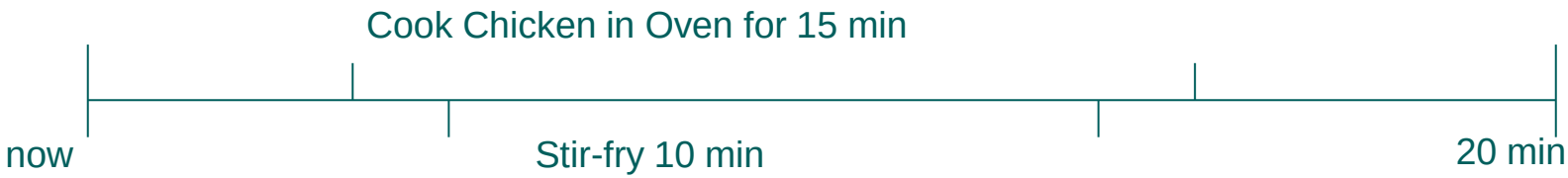
Code: Specification

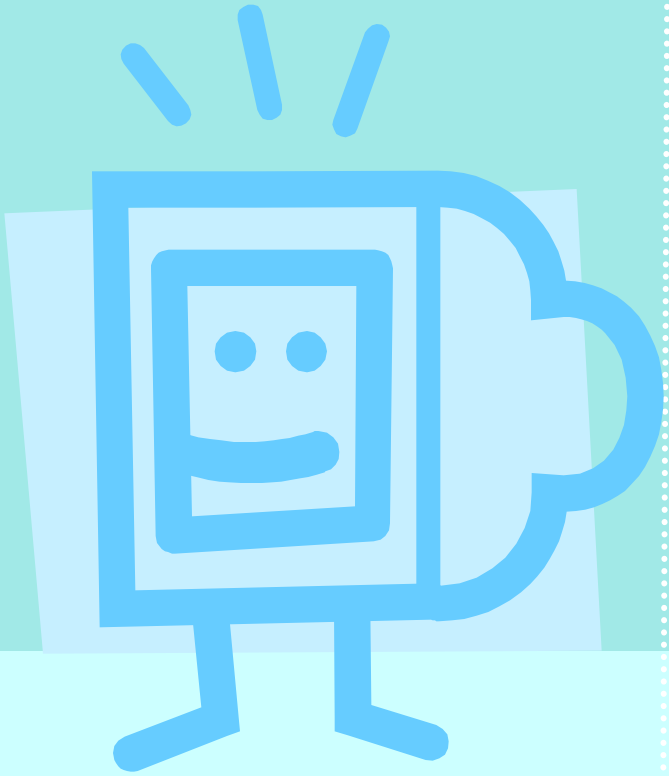


```
process cooking_robot;
 call-port RangeOn [deadline 2 sec], RangeOff [deadline 2 sec];
           /* Fry */
 in-port: OvenOn [deadline 2 sec], OvenOff [deadline 2 sec];
           /* Chicken */
 var ToBeDone : time
begin
   start now within 20 min do
       call (RangeOn, nil);
       send (OvenOn, 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;
```

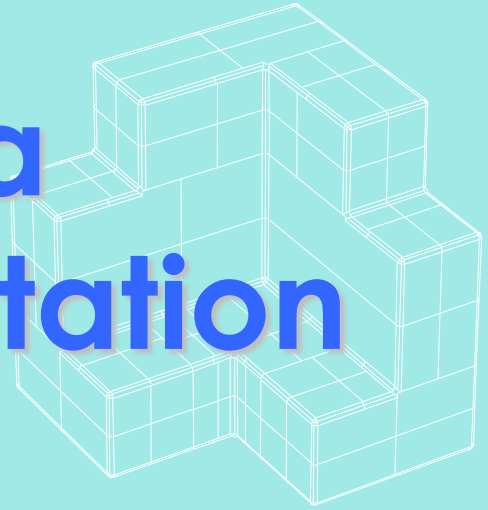
System View


Order
Food
Oven
Chicken
(In-Port)
Fries
(Call-Port)
Range
On
Off
On
Off
Cooking System


Cook Chicken in Oven for 15 min
now
Stir-fry 10 min
20 min



Code: Ada Implementation



```
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 time;
                      prd_4_b: in time;
                      prd_4_f: in time);
      ENTRY serve_with_food();
   END Controller;

END CRS;
```

```
TASK Oven IS
   ENTRY bake(prd: in time);
END Controller;
```

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

```
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
         ACCCEPT sereve_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 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;
```

Ada Package View