Content(2)

- **Object-oriented Software Development Methodology**
  - Outline of Unified Process and Use-case Driven Approach
  - Elevator Control System:
    Problem Description and Use-case Model
  - Elevator Control System:
    Finding of Problem Domain Objects
  - Elevator Control System:
    Sub-System Design and Task Design
  - Elevator Control System:
    Performance Evaluation
- **Product Line Technology**
  - Feature modeling
- **Aspect Oriented Software Design**
- **Contribution of OOT in Software Engineering**
  - History of SE Technologies and Contribution of OOT in SE field

Software Product Line Technologies

Hassan Gomaa, Designing Software Product Line with, Addison Wesley, (2004)

Koichiro Ochimizu

School of Information Science
Japan Advanced Institute of Science and Technology

Basic Ideas and Terms

- **A software product line** is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of **core assets** in prescribed way.
- **Core assets** are those asset that form the basis for the software product line.
  - Core assets often include, but are not limited to, the architecture, reusable software components, domain models, requirement statements, documentations and specifications, performance models, schedules, budgets, test plans, test cases, work plans, and process description

Paul Clements an Linda Northrop, "Software Product Lines", Addison-Wesley, 2002.

What is Software Development?

- **Development** is a generic term used to describe how core assets come to fruition.
  - The organization can **Build** it itself (either from scratch or by mining legacy software),
  - **Purchase it** (buy it, largely unchanged, off the shelf)
  - **Commission it** (contract with someone else to develop it especially for the organization)
- Development may actually involve building, acquisition, purchase, retrofitting earlier work.

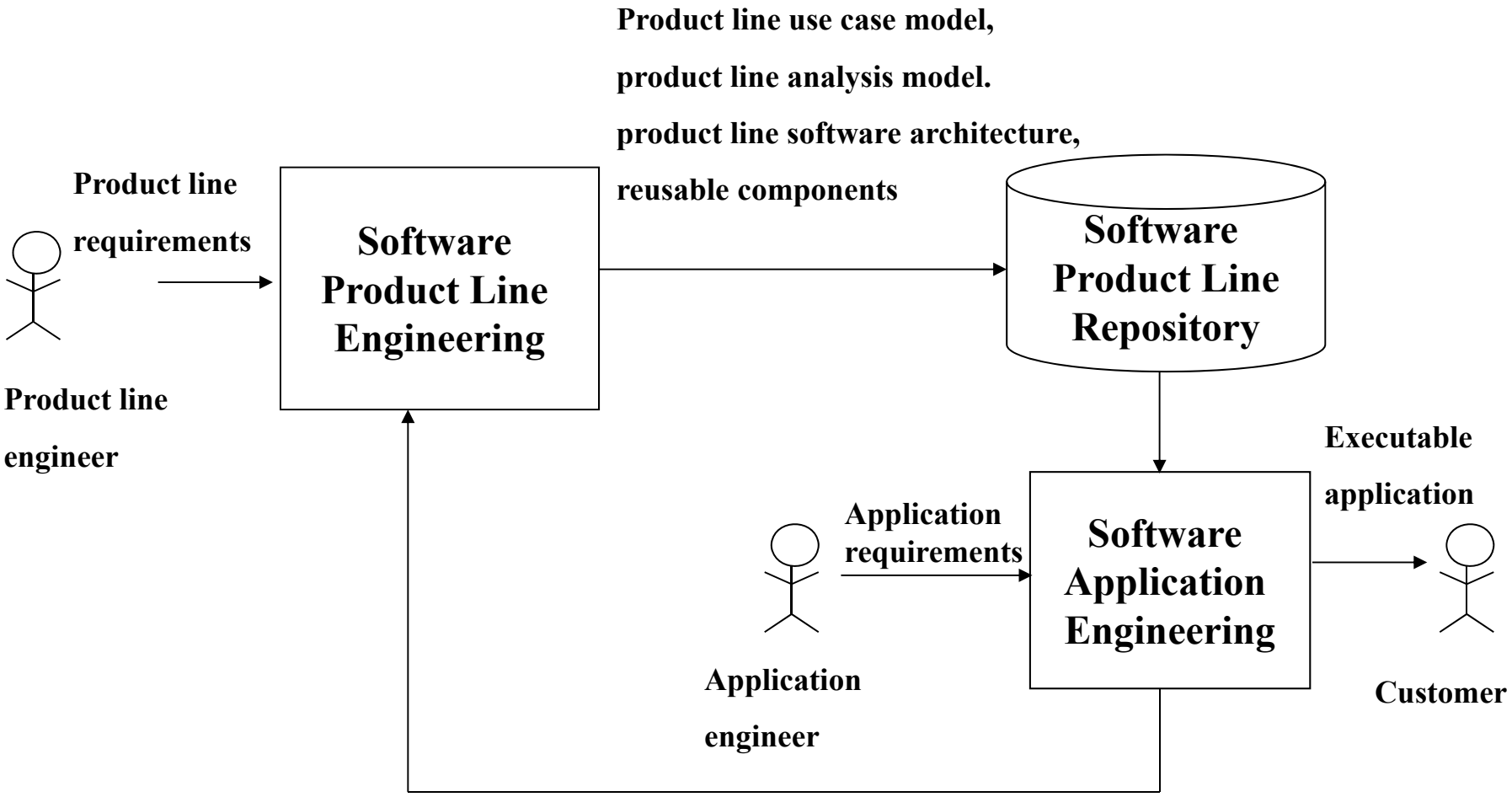
Paul Clements an Linda Northrop, "Software Product Lines", Addison-Wesley, 2002.

Basic Ideas and Terms(2/2)

- **Software product line practice** is the systematic use of core assets to assemble, instantiate, or generate the multiple products that constitute a software product line. Software Product line practice involves strategic, large-grained reuse.
- **The Three Essential Activities**
  - Core Asset Development (or Domain Engineering)
  - Product Development using the core assets (Application Engineering)
  - Management

Paul Clements an Linda Northrop, "Software Product Lines", Addison-Wesley, 2002.

PLUS
(Product Line UML-based Software Engineering)



Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Characteristics of PLUS

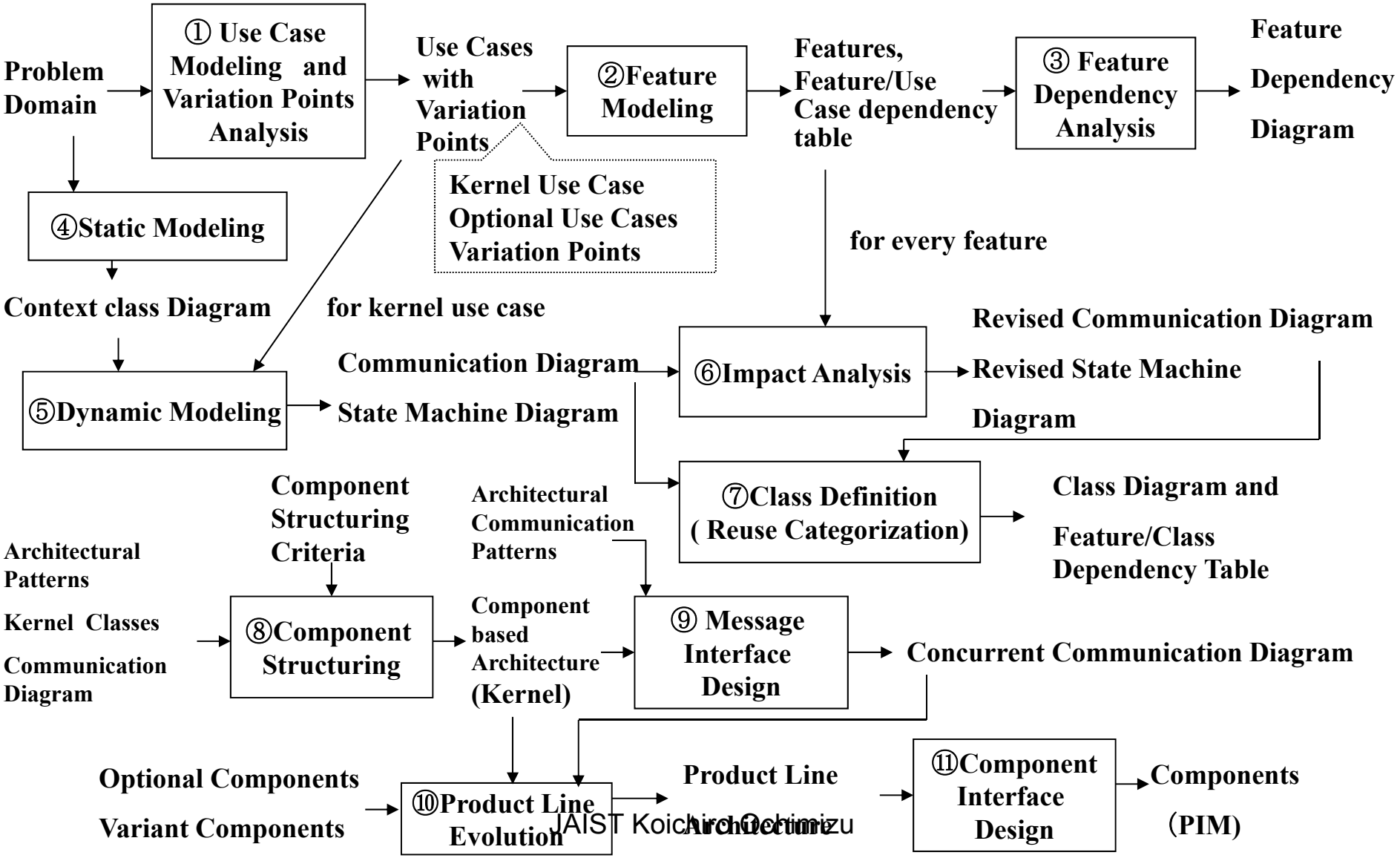
- Feature modeling based on Use case modeling with variation points.
- Use case driven object-oriented approach , partially adopting CBSD(Component-Based Software
- Definite correspondence among Features, Classes and Components

What is a Feature?

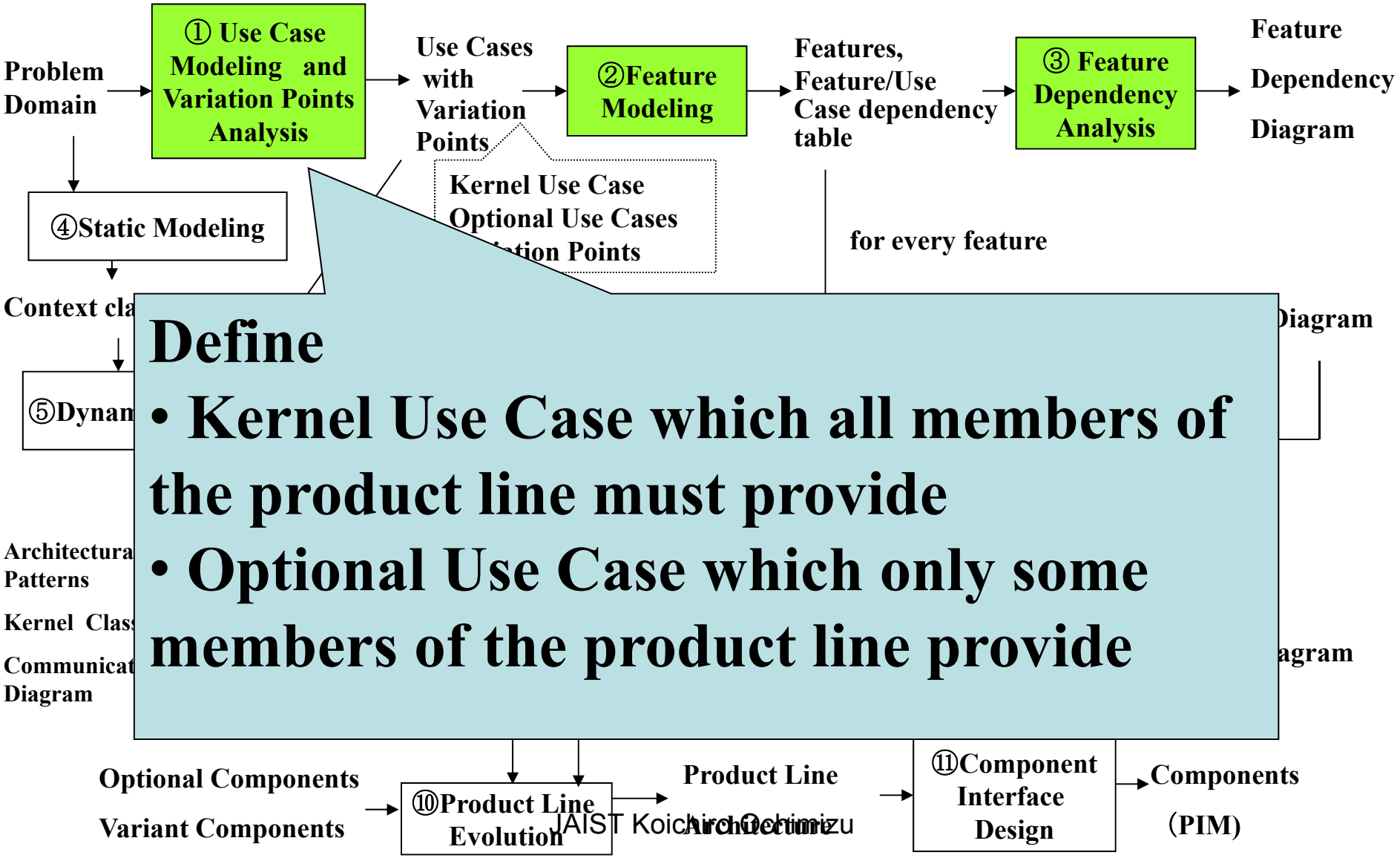
- Features are characteristics that are used to differentiate among members of the product line
- Feature Modeling is a Variability Analysis in requirement modeling to determine and define the common and variable functionality of a software product line

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

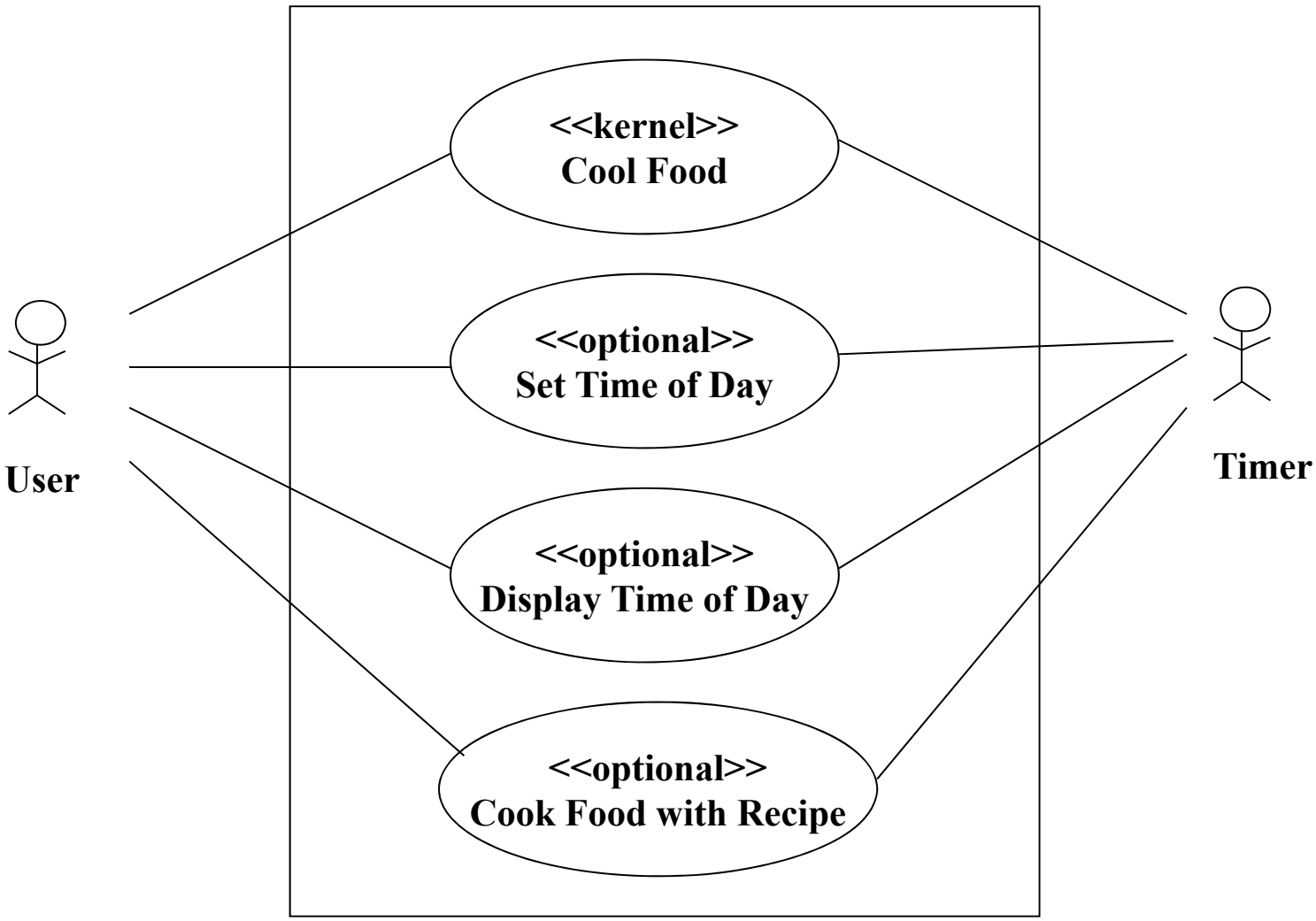
PLUS Phases in Product line Engineering



Outline of PLUS (Use Case Modeling and Variation Points Analysis)



Use Case Model



Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Kernel Use Case “ Cook Food ”

- **Use case name:** Cook Food
- **Reuse category:** Kernel
- **Summary:** User puts food in oven, and microwave oven cooks food.
- **Actors:** User(primary), Timer(secondary)
- **Precondition:** Microwave oven is idle
- **Description:**
  1. User opens the door, puts food in the oven, and closes the door.
  2. User presses the **Cooking Time** button.
  3. System prompts for cooking time.
  4. User enters the cooking time on the numeric keypad and presses **Start**.
  5. System starts cooking the food.
  6. System continually displays the cooking time remaining.
  7. Timer elapses and notifies the system.
  8. System stops cooking the food and displays the end message.
  9. User opens the door, removes the food from the oven, and closes door.
  10. System clears display.

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Kernel Use Case “ Cook Food ”

- **Alternatives:**
  - Line1: User presses **Start** when the door is open. System does not start cooking.
  - Line4: User presses **Start** when the door is closed and the oven is empty. System does not start cooking.
  - Line4: User presses **Start** when the door is closed and the cooking time is equal to zero. System does not start cooking.
  - Line6: User opens the door during cooking. System stops cooking. User removes food and presses **Cancel,** or user closes the door and presses **Start** to resume cooking.
  - Line6: User presses **Cancel.** System stops cooking. User may press **Start** to resume cooking. Alternatively, user may press **Cancel** again; system then cancels timer and clears display.
- **Post-conditions:** Microwave oven has cooked the food.

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Variability Analysis

- Define variation points for each use case
- For small variations, the variation point is described in the (kernel) use case itself, identifying the place in the use case where the change occur
- For large variations, the variation is defined as an optional use case
- An optional use case has its own variation points

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Variation Points in the "Cook Food" Use Case(1/3)

- **Name:** Display Language
- **Type of functionality:** Mandatory alternative
- **Line number(s):** 3,8
- **Description of functionality:** There is a choice of language for displaying messages. The default is English. Alternative mutually exclusive languages are French, Spanish, German, or Italian.

- **Name:** Weight Sensor
- **Type of functionality:** Mandatory alternative
- **Line number(s):** 1
- **Description of functionality:** Cooking is prohibited if no item is present. The default is Boolean weight sensor, which indicates if item is present. Alternative mutually exclusive variation is analog sensor. Analog weight sensor provides weight of item.

- **Name:** Heating Element
- **Type of functionality:** Mandatory alternative
- **Line number(s):** 5
- **Description of functionality:** Default is a one-level heating element: high power level. Alternative is a multi-level heating element, with high, medium, and low power levels.

Hassan Gomaa, "Designing Software Product Lines with UML", Addison-Wesley, 2005.

Variation Points in the "Cook Food" Use Case(2/3)

- **Name:** Power Level
- **Type of functionality:** Optional
- **Line number(s):** 2
- **Description of functionality:** Microwave oven has power level buttons for high power(default), medium , and low. User may select the power level. Requires multi-level heating element as prerequisite.

- **Name:** Display Unit
- **Type of functionality:** Mandatory alternative
- **Line number(s):** 3, 4, 6, 8, 10
- **Description of functionality:** Default is a one-line display unit. Alternative is multi-line display unit.

- **Name:** Minute Plus
- **Type of functionality:** Optional
- **Line number(s):** 2, 6
- **Description of functionality:** User may press **Minute Plus**, which results in one minute being added to the cooking time. If the cooking time was previously zero, cooking is started

Hassan Gomaa, "Designing Software Product Lines with UML", Addison-Wesley, 2005.

Variation Points in the “Cook Food” Use Case(3/3)

- **Name:** Light
- **Type of functionality:** Optional
- **Line number(s):** 1, 5, 8, 9
- **Description of functionality:** If light option is selected, lamp is switched on for duration of cooking and when the door is open. Light is switched off when door is closed and when cooking stops.

- **Name:** Turntable
- **Type of functionality:** Optional
- **Line number(s):** 5,8
- **Description of functionality:** If turntable option is selected, turntable rotates for duration of cooking.

- **Name:** Beeper
- **Type of functionality:** Optional
- **Line number(s):** 8
- **Description of functionality:** If beeper option is selected, system activates the beeper when cooking stops.

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Optional Use Case "Set Time of Day"

- **Use case name:** Set Time of Day
- **Reuse category:** Optional
- **Dependency:** Variation point in Cook Food use case: at Display Unit variation point, select Multi-line Display.
- **Summary:** User sets time-of-day clock
- **Actors:** User
- **Precondition:** Microwave oven is idle
- **Description:**
  1. User presses **Time of Day(TOD)** button.
  2. System prompts for time of day.
  3. User enters the time of day (in hours and minutes) on the numeric keypad.
  4. System stores and displays the entered time of day.
  5. User presses **Start**.
  6. System starts the time-of-day timer.
- **Alternatives:**

  Line 1,3: If the oven is busy, the system will not accept the user input.

  Line 5: The user may press Cancel if the incorrect time was entered. The system clears the display.
- **Variation Points in the Set Time of Day Use case**
  - **Name:** 12/24Hour Clock
  - **Type of functionality:** Mandatory alternative.
  - **Line number(s):** 4
  - **Description of functionality:** TOD display is either 12-hour clock(default) or 24-hour clock.
- **Post Condition:** TOD clock has been set.

Hassan Gomaa, "Designing Software Product Lines with UML", Addison-Wesley, 2005.

Optional Use Case "Display Time of Day"

- **Use case name:** Display Time of Day
- **Reuse category:** Optional
- **Dependency:** Variation point in Cook Food use case: at Display Unit variation point, select Multi-line Display.
- **Summary:** System displays time-of-day
- **Actors:** Timer(primary actor), User(secondary actor).
- **Precondition:** TOD clock has been set(by Set Time of Day use case)
- **Description:**
  1. Timer notifies system that one second has elapsed.
  2. System increments TOD clock every second, adjusting for minutes and hours.
  3. System updates the display with time of day every minute.
- **Variation Points in the Display Time of Day Use case**
  - **Name:** 12/24Hour Clock
  - **Type of functionality:** Mandatory alternative.
  - **Line number(s):** 2
  - **Description of functionality:** TOD display is either 12-hour clock(default) or 24-hour clock.
- **Post Condition:** TOD clock has been updated (every second) and time of day displayed(every minute).

Hassan Gomaa, "Designing Software Product Lines with UML", Addison-Wesley, 2005.

Optional Use Case “ Cook Food with Recipe”(1/3)

- **Use case name:** Cook Food with Recipe
- **Reuse category:** Optional
- **Dependency:** Variation points in Cook Food use case: at Display Unit variation point, select Multi-line Display; at Heating Element variation point, select Multi-level Heater; at Weight Sensor variation point, select Analog Weight Sensor.
- **Summary:** User puts food in microwave oven cooks food, using recipe.
- **Actors:** User(primary), Timer(secondary)
- **Precondition:** Microwave oven is idle
- **Description:**
  1. User opens the door, puts food in the oven, and closes the door.
  2. User presses the desired recipe button from the recipe buttons on the keypad.
  3. System displays the recipe name. Recipe has name, power level(p), fixed time(t1), and time per unit weight(t2).
  4. User presses **Start** button.
  5. System starts cooking the food for a time given by the following equation: Cooking Time = t1 + w*t2, where t1 and t2 are times specified in the recipe.
  6. System continually displays the cooking time remaining.
  7. Timer elapses and notifies the system.
  8. System stops cooking the food and displays the end message.
  9. User opens the door, removes the food from the oven, and closes door.
  10. System clears display.

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Optional Use Case “ Cook Food with Recipe”(2/3)

- **Alternatives:**

  Line1: User presses **Start** when the door is open. System does not start cooking.

  Line4: User presses **Start** when the door is closed and the oven is empty. System does not start cooking.

  Line4: User presses **Start** when the door is closed and a recipe has not been chosen. System does not start cooking.

  Line4: User presses **Cancel**. System cancels recipe and clears display.

  Line6: User opens the door during cooking. System stops cooking. User removes food and presses **Cancel,** or user closes the door and presses **Start** to resume cooking.

  Line6: User presses **Cancel.** System stops cooking. User may press **Start** to resume cooking. Alternatively, user may press **Cancel** again; system then cancels the recipe and clears display.

  Line7: If the recipe has more than one step, system completes one step, cooking food for the computed time and specified power level, and then proceeds to the next step.

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Optional Use Case "Cook Food with Recipe" (3/3)

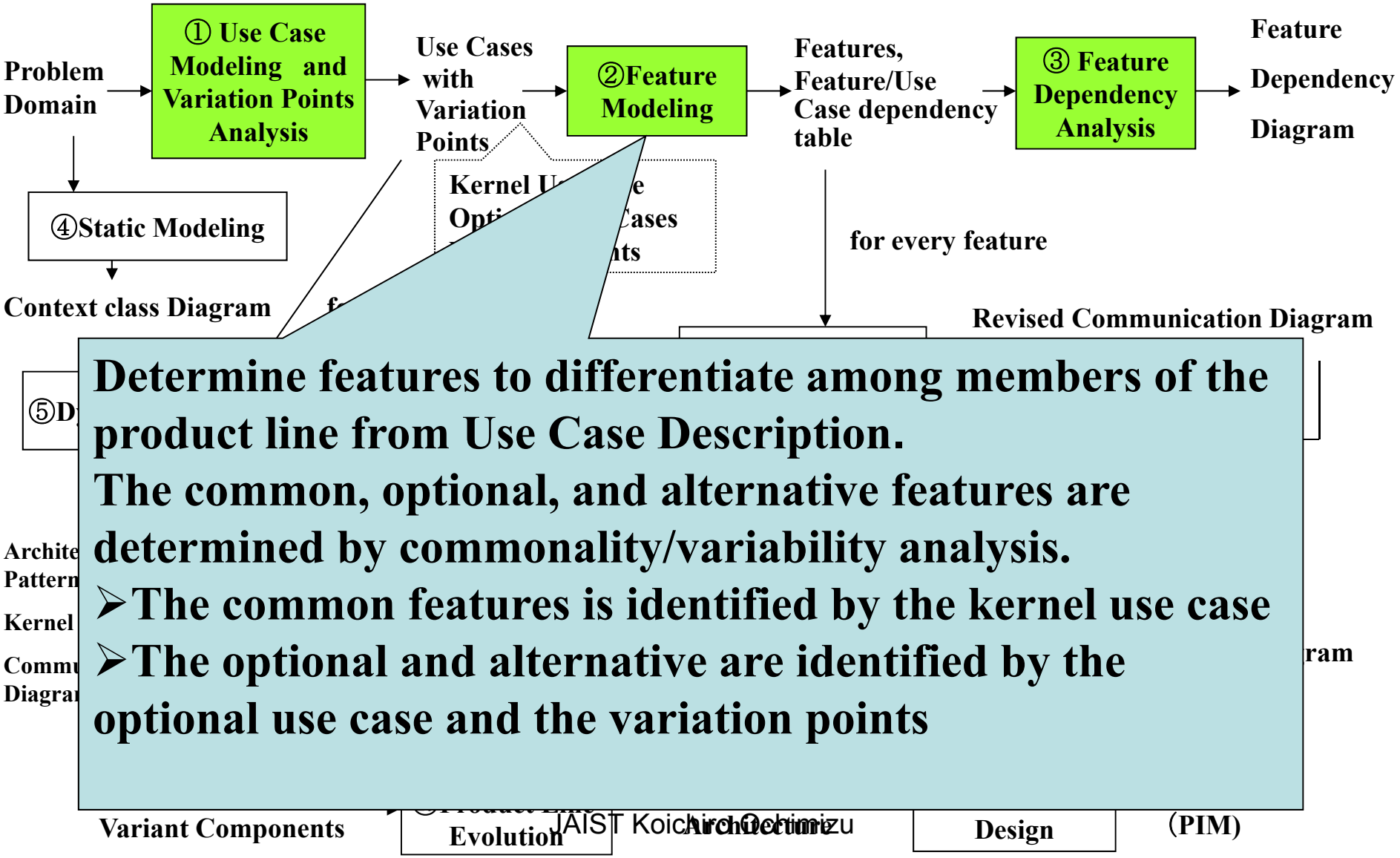
- **Name:** Display Language
- **Type of functionality:** Mandatory alternative
- **Line number(s):** 3,8
- **Description of functionality:** There is a choice of language for displaying messages. The default is English. Alternative mutually exclusive languages are French, Spanish, German, or Italian.

- **Name:** Light
- **Type of functionality:** Optional
- **Line number(s):** 1, 5, 8, 9
- **Description of functionality:** If light option is selected, lamp is switched on for duration of cooking and when the door is open. Light is switched off when door is closed and when cooking stops.

- **Name:** Turntable
- **Type of functionality:** Optional
- **Line number(s):** 5,8
- **Description of functionality:** If turntable option is selected, turntable rotates for duration of cooking.

- **Name:** Beeper
- **Type of functionality:** Optional
- **Line number(s):** 8
- **Description of functionality:** If beeper option is selected, system activates the beeper when cooking stops.
- **Post-condition:** Microwave oven has cooked the food using recipe

Outline of PLUS (Feature Modeling)



Feature Modeling

- After the use case model, the next step is to address is the feature model and to determine how the use case and use case variation points correspond to features.
- The feature model is developed as a result of a commonality/variability analysis in which the common, optional, and alternative features are determined.
  - The common features identify the common functionality in the product line , as specified by the kernel use case
  - The optional and alternative features represent the variability in the product line as specified by the optional use case and the variation points

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Feature/Use Case Dependencies(1/3)

Feature Name	Feature Category	Use Case Name	Use Case Category/ Variation Point (vp)	Variation Point Name
Microwave Oven Kernel	common	Cook Food	Kernel	
Light	optional	Cook Food	VP	Light
Turn Table	optional	Cook Food	VP	Turn Table
Beeper	optional	Cook Food	VP	Beeper
Minute Plus	optional	Cook Food	VP	Minute Plus
One-line Display	default	Cook Food	VP	Display Unit
Multi-line Display	alternative	Cook Food	VP	Display Unit

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Feature/Use Case Dependencies(2/3)

Feature Name	Feature Category	Use Case Name	Use Case Category/ Variation Point (vp)	Variation Point Name
English	default	Cook Food	vp	Display Language
French	alternative	Cook Food	vp	Display Language
Spanish	alternative	Cook Food	vp	Display Language
German	alternative	Cook Food	vp	Display Language
Italian	alternative	Cook Food	vp	Display Language
Boolean Weight	default	Cook Food	vp	Weight Sensor
Analog Weight	alternative	Cook Food	vp	Weight Sensor

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

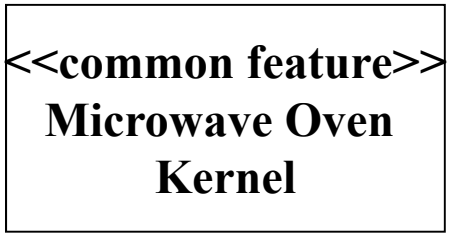
Feature/Use Case Dependencies(3/3)

Feature Name	Feature Category	Use Case Name	Use Case Category/ Variation Point (vp)	Variation Point Name
One-level Heating	default	Cook Food	vp	Heating Element
Multi-level Heating	alternative	Cook Food	vp	Heating Element
Power Level	optional	Cook Food	vp	Power Level
TOD Clock	optional	Set Time of Day	optional	
		Display Time of Day	optional	
12/24 Hour Clock	parameterized	Set Time of Day	vp	12/24 Hour Clock
		Display Time of Day		
Recipe	optional	Cook Food with Recipe	optional	

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Every Microwave Oven must have

- **Door**
  - Every microwave oven has a door. Cooking is permitted only when the door is closed.
- **Weight Sensor**
  - Every microwave oven has a weight sensor. Cooking is permitted only when there is an item in the oven, as detected by the weight sensor
- **Keypad**
  - The basic keypad consists of a numeric keypad for entering the time, a **Cooking Time** button, a **Start** button, and **Stop**/**Cancel** button.
- **Display**
  - Every microwave oven has a display to show the time remaining, as well as any prompts or warning messages.
- **Heating element**
  - This is the power source for cooking food.
- **Timer**
  - A timer is needed to count down the cooking time remaining and to determine when cooking must be stopped



Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

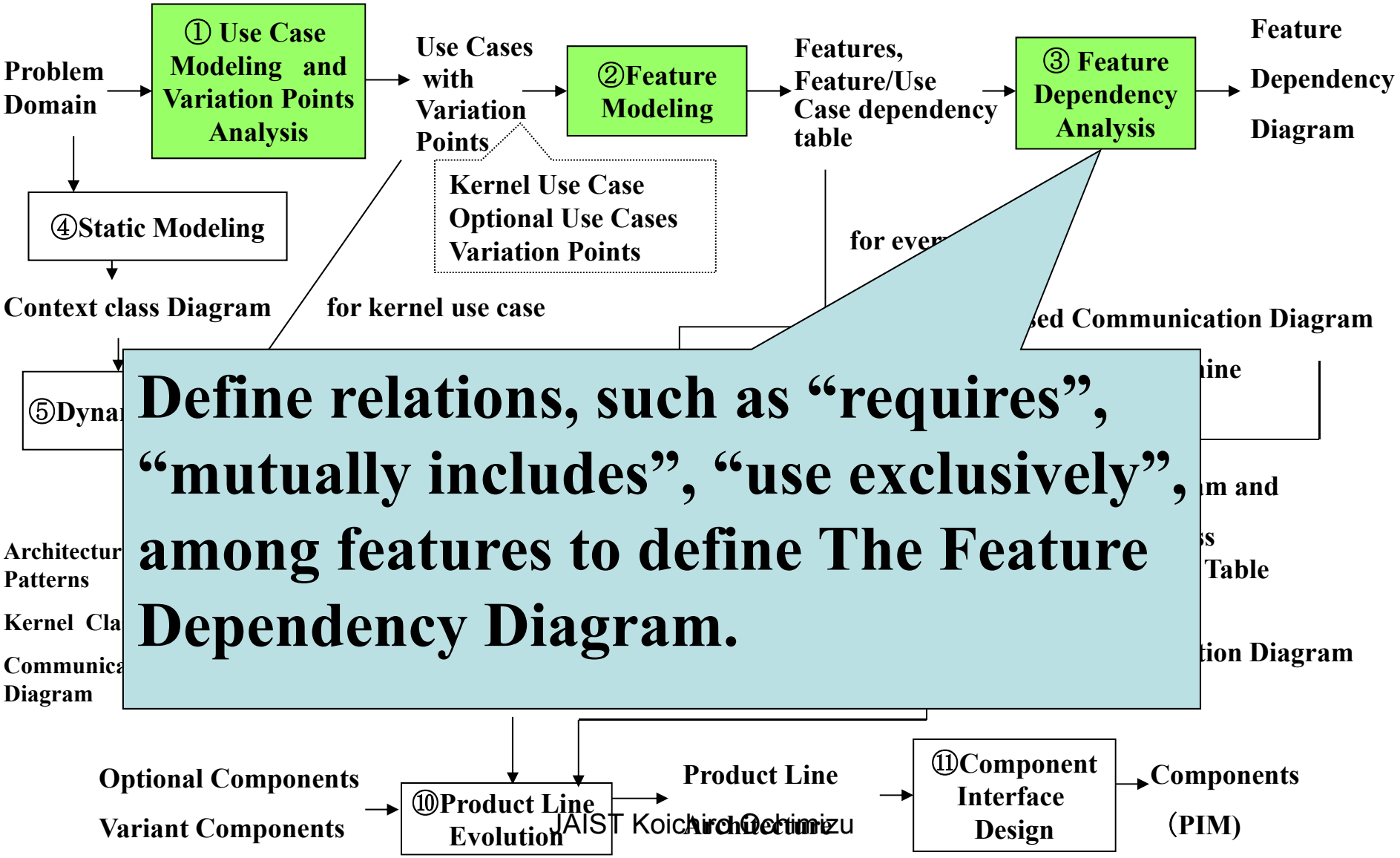
Optional Features

- <<**optional feature**>>Light
  - If the light option is selected, the lamp is switched on for the duration of cooking and when the door is open.
- <<**optional feature**>> Turntable
  - If the turn table option is selected, the turn table rotates for the duration of cooking.
- <<**optional feature**>> Beeper
  - If the beeper option is selected, the system activates the beeper when cooking stops.
- <<**optional feature**>> Minute Plus
  - If the oven is already cooking food, pressing **Minute Plus** button adds one minute to the cooking time
  - If the oven is not cooking food and the cooking time is set to zero, the cooking time is set to 60 seconds and cooking is started.
  - If the door is open, a press of **Minute Plus** button is ignored.
  - If the oven is not cooking food but the cooking time is greater than zero, a press of the Minute Plus button is ignored.

<<optional feature>> Light	<<optional feature>> Turntable	<<optional feature>> Beeper	<<optional feature>> Minute Plus

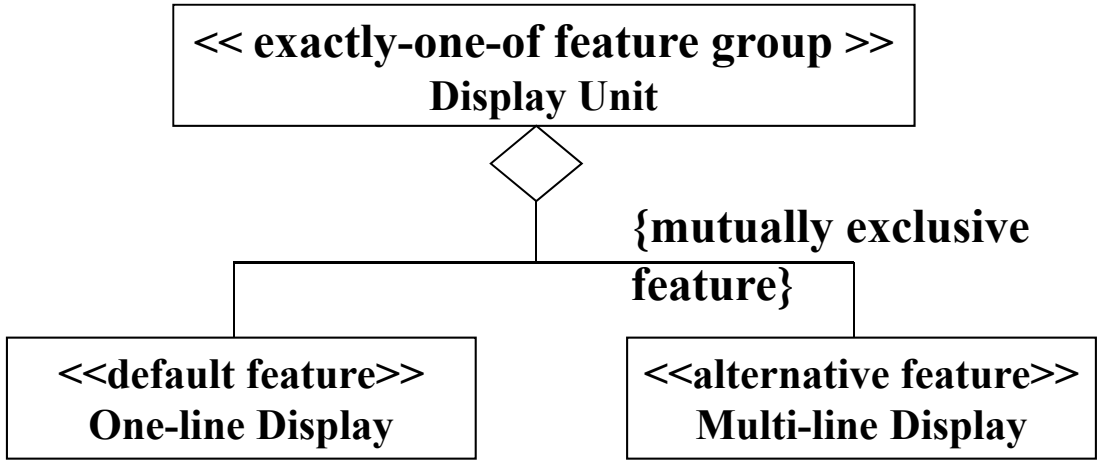
Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Outline of PLUS (Feature Dependency)



Alternative Features and Feature Groups

- <<**exactly-one-of feature group**>> Display Unit
  {default = One-line Display, alternative = Multi-line Display}
- << **exactly-one-of feature group** >> Display Language
  {default = English, alternative = French, Spanish, German, Itakian}
- << **exactly-one-of feature group** >> Weight
  {default = Boolean Weight, alternative = Analog Weight}
- << **exactly-one-of feature group** >> Heating Element
  {default = One-level Heating, alternative = Multi-level Heating}
  - The default heating element mode is one level: high. The alternative is multi-level, with high, medium, and low power levels.



Hassan Gomaa, "Designing Software Product Lines with UML", Addison-Wesley, 2005.

Optional Features with Prerequisite and Mutually inclusive Features

- **Power Level**

  <<optional feature>> Power Level {mutually includes = Multi-level Heating}

  – With this optional feature, a **Power Level** button is provided on the keypad. The power level can be set to high, medium, or low. One consideration is whether his functionality should be part of Multi-level Heating feature. However, multi-level heating can be provided for cooking with recipes, in which case it can be used without a **Power Leve**l button. For this reason, Power Level and Multi-level Heating are kept as separate but mutually inclusive features.

- **Recipe**

  <<optional feature>> Recipe { prerequisite = Multi-line Display, mutually includes = Analog Weight, Multi-level Heating }

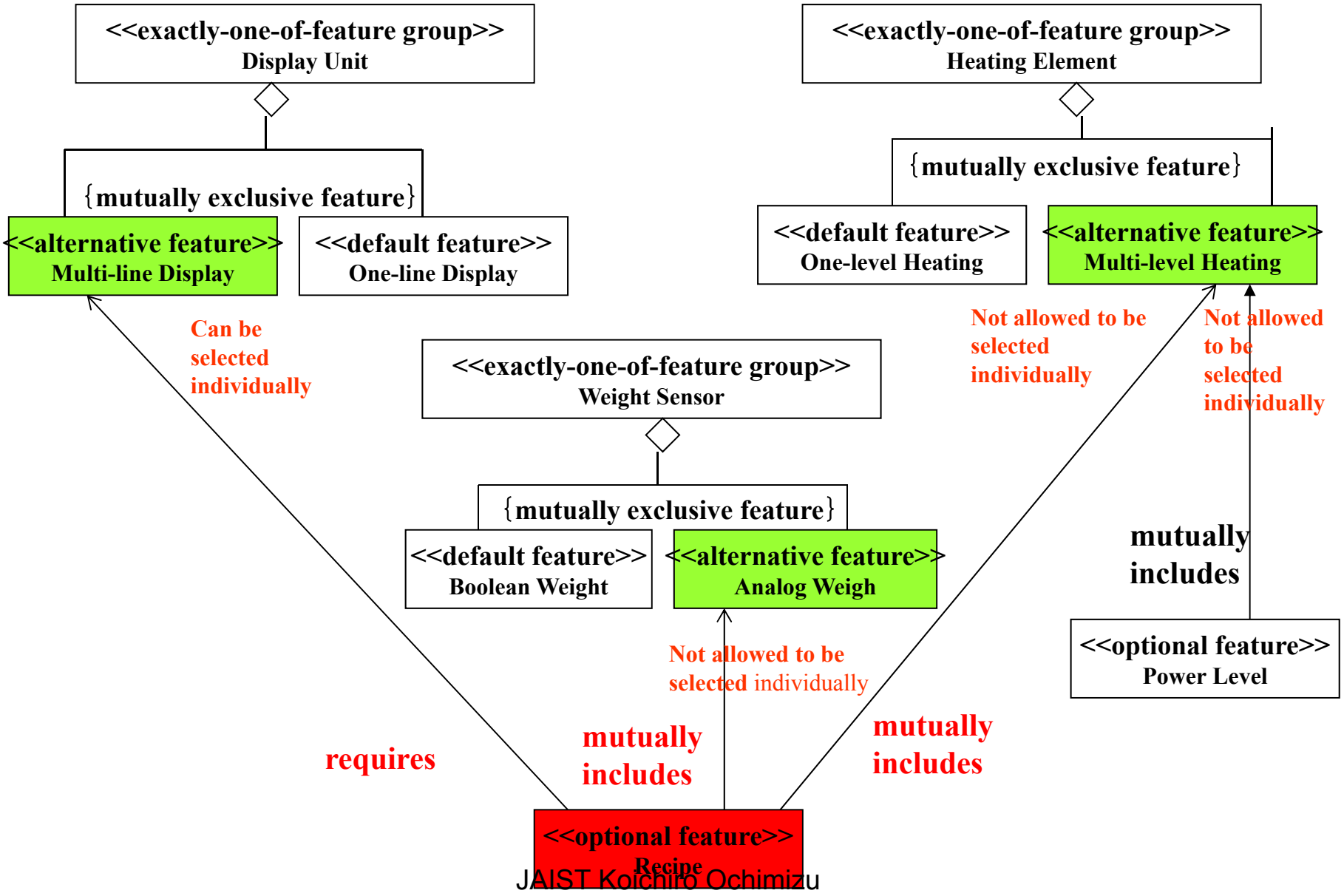
  – With this optional feature, a **Recipe** button is provided on the keypad. Food is cooked as prescribed in the selected recipe. There is one prerequisite feature: Multi-line Display. There are two mutually inclusive features: Analog Weight(which is used only for cooking with recipes) and Multi-level Heating (as described above):

- **TOD Clock**

  <<optional feature>> TOD Clock { prerequisite = Multi-line Display}

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Difference of meaning between “requires” and “mutually includes”



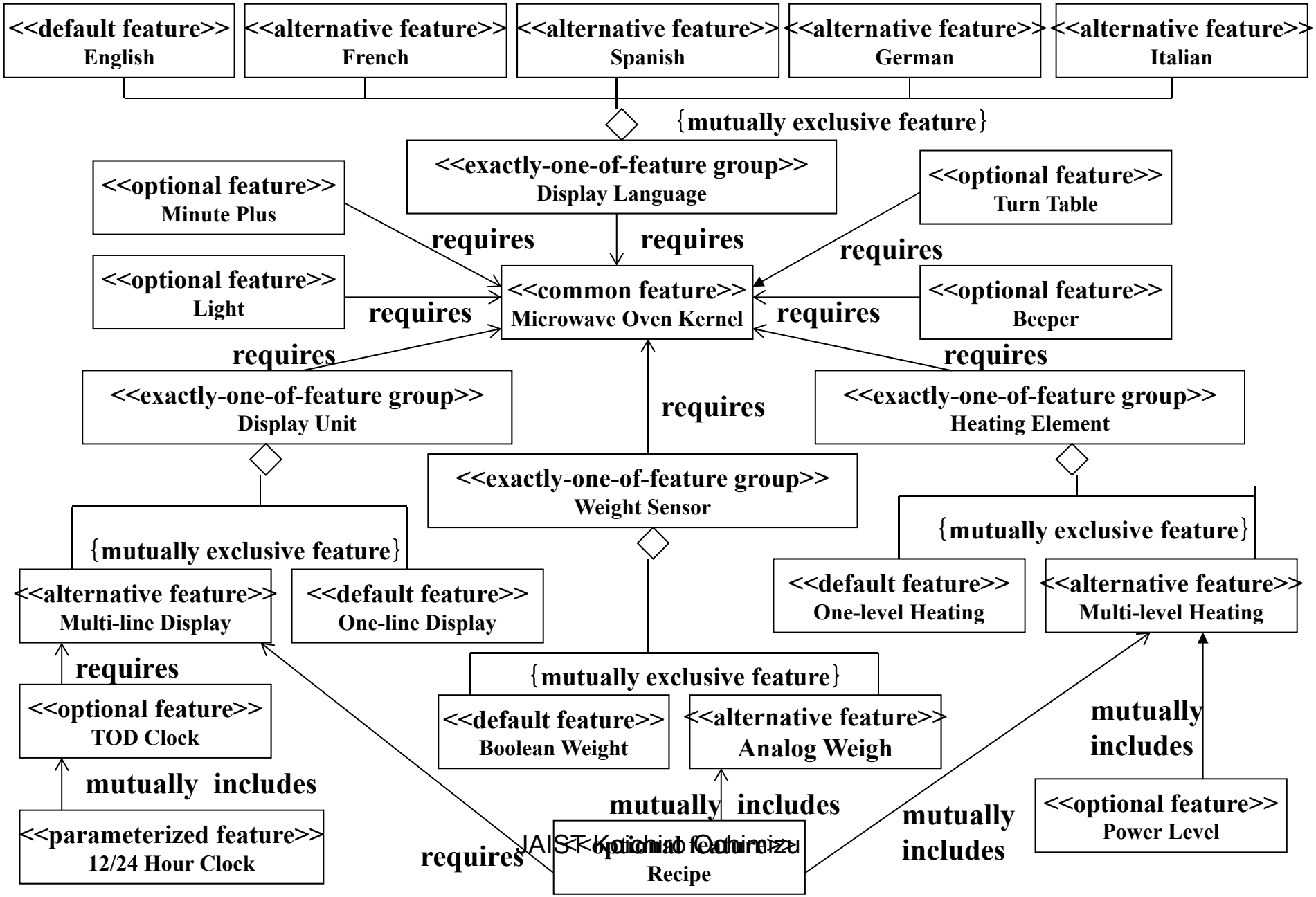
Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

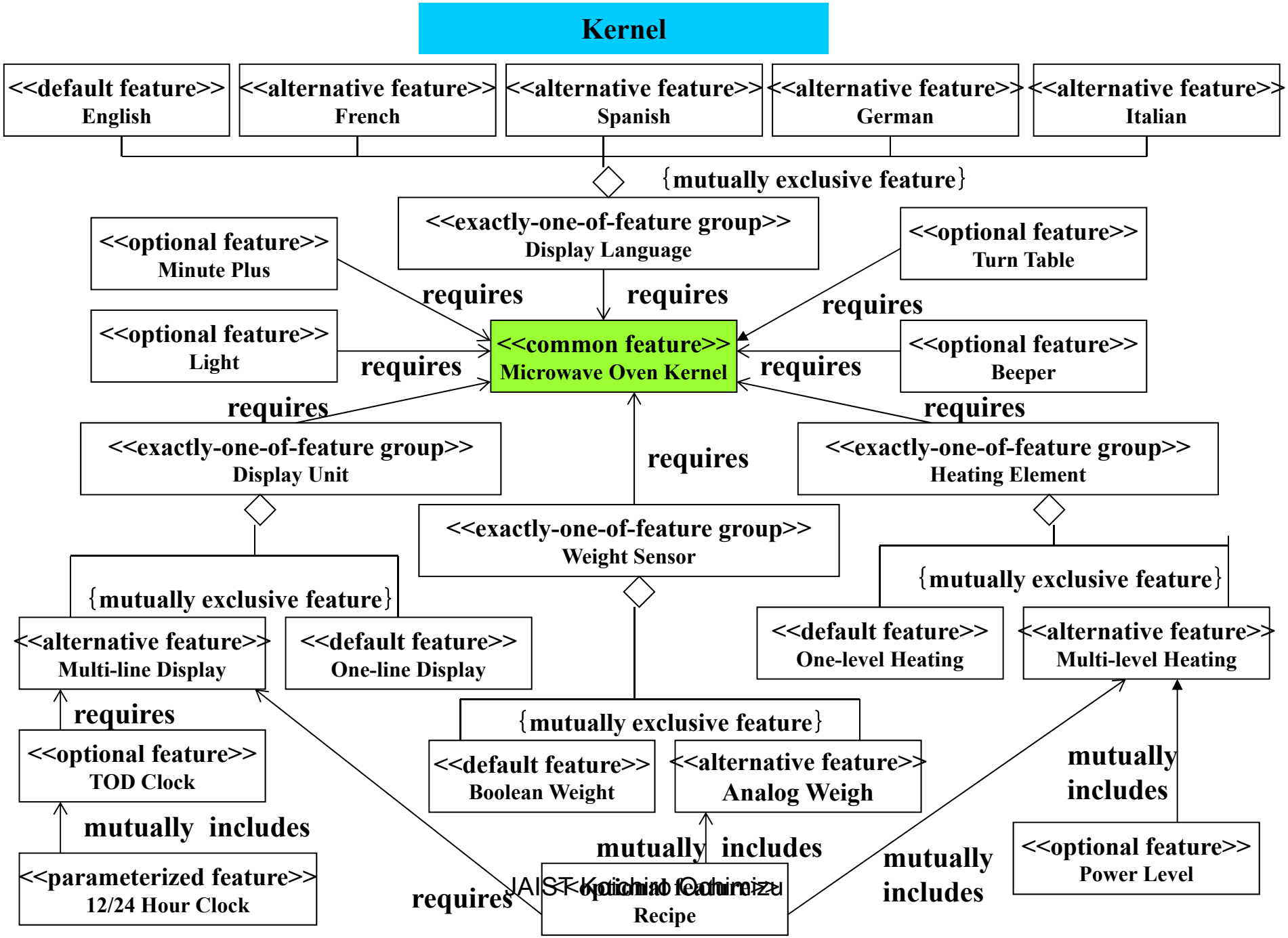
Parameterized Features

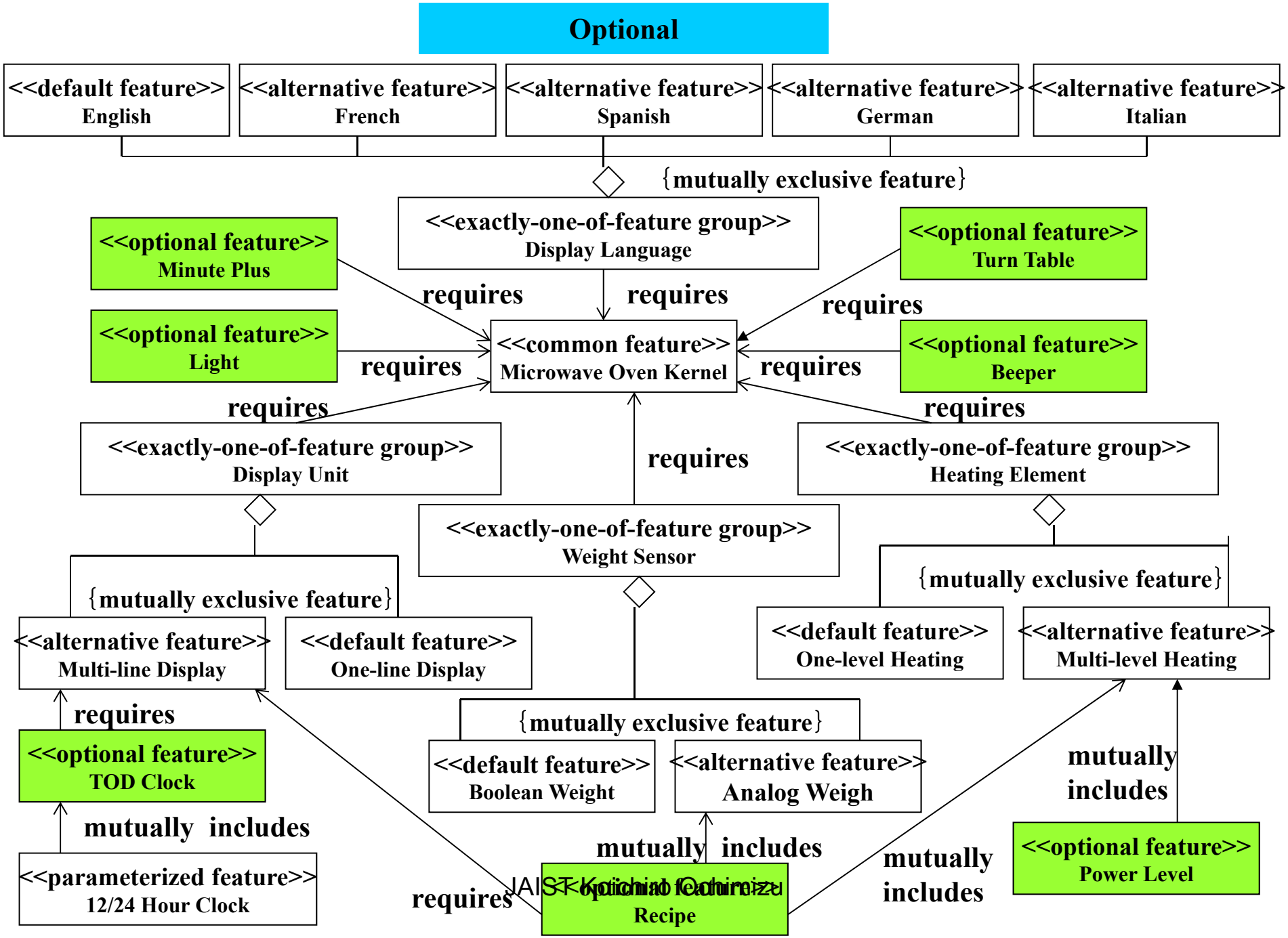
- **12/24 Hour Clock**
  <<parameterized feature>> 12/24 Hour Clock
  { type = Time, permitted value = 12:00, 24:00,
  default value = 12:00, mutually includes = TOD Clock}

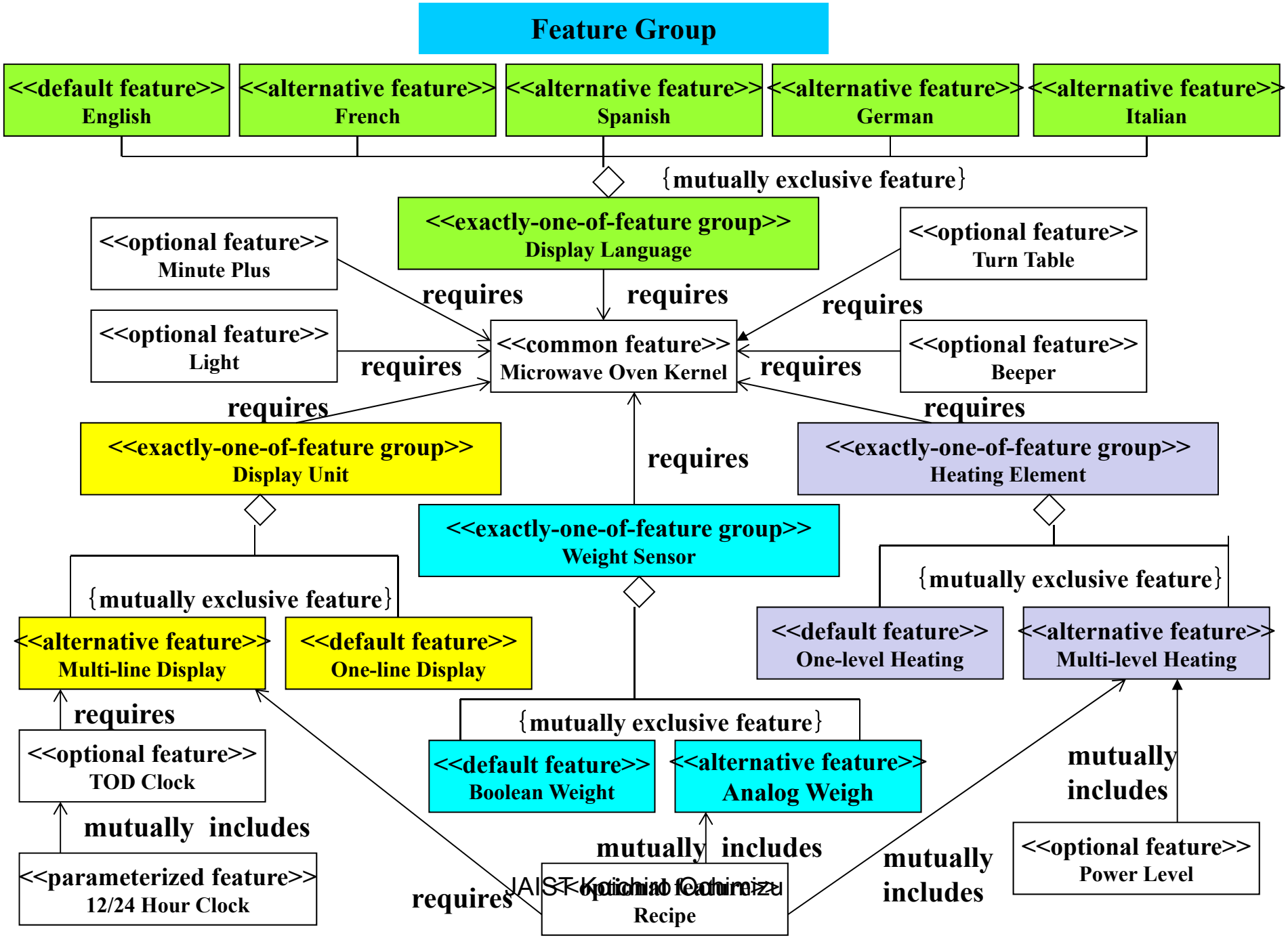
Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

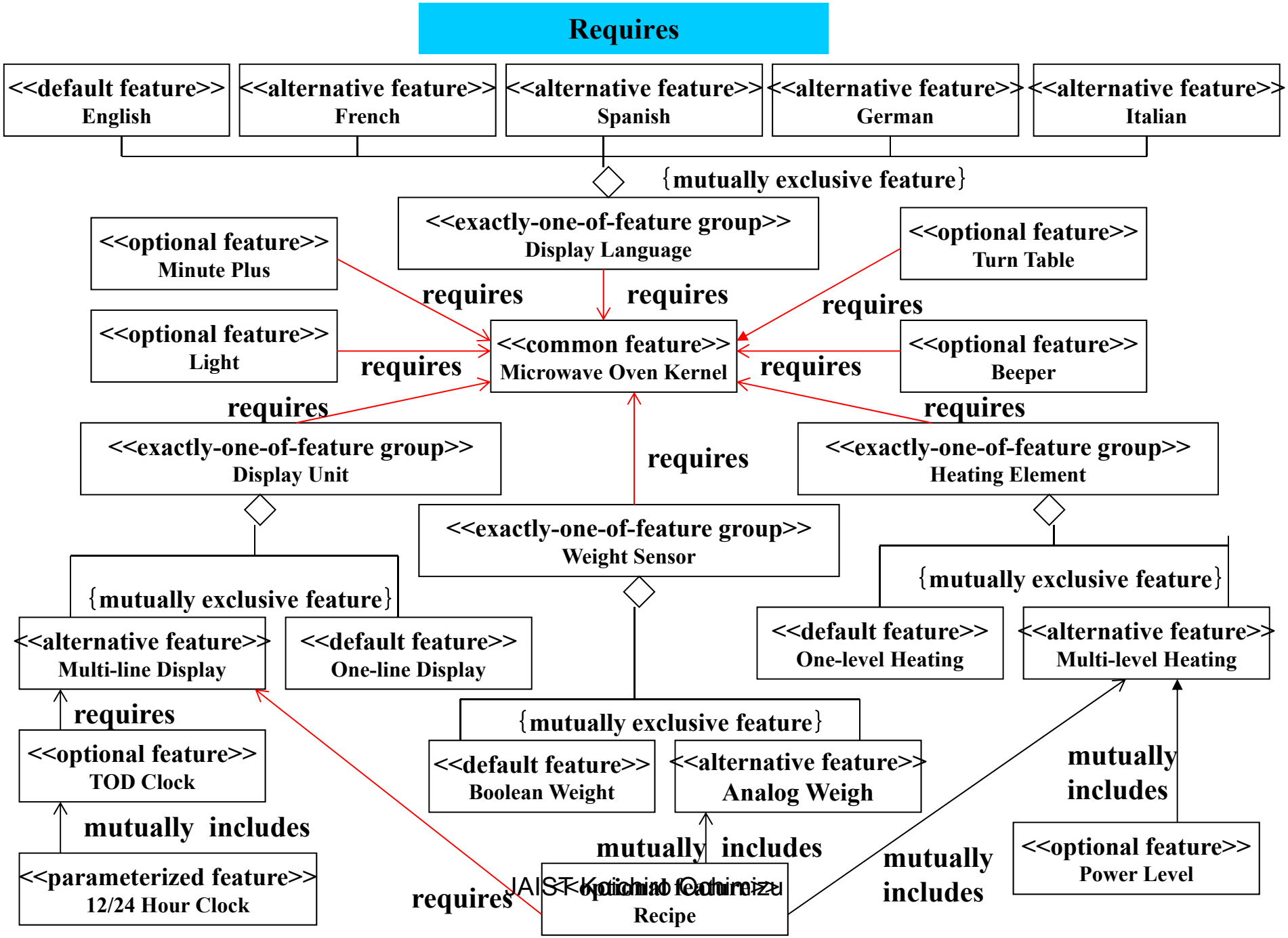
Feature Dependency Diagram

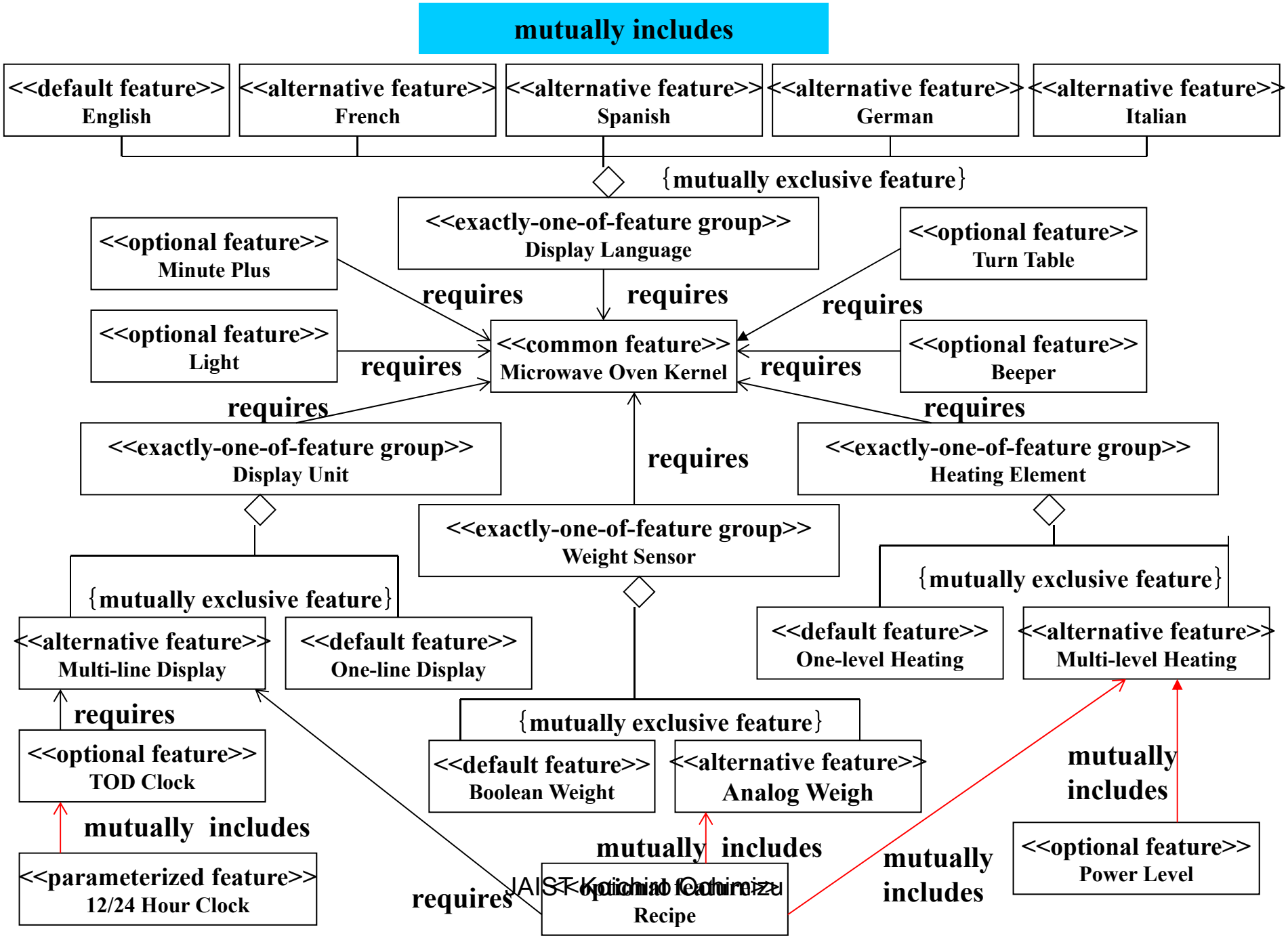



Kernel
<<default feature>>
English
<<alternative feature>>
French
<<alternative feature>>
Spanish
<<alternative feature>>
German
<<alternative feature>>
Italian
{mutually exclusive feature}
<<exactly-one-of-feature group>>
Display Language
<<optional feature>>
Minute Plus
<<optional feature>>
Turn Table
<<optional feature>>
Light
<<common feature>>
Microwave Oven Kernel
<<optional feature>>
Beeper
requires
<<exactly-one-of-feature group>>
Display Unit
<<exactly-one-of-feature group>>
Weight Sensor
<<exactly-one-of-feature group>>
Heating Element
{mutually exclusive feature}
<<alternative feature>>
Multi-line Display
<<default feature>>
One-line Display
<<default feature>>
One-level Heating
<<alternative feature>>
Multi-level Heating
<<optional feature>>
TOD Clock
<<default feature>>
Boolean Weight
<<alternative feature>>
Analog Weigh
<<optional feature>>
Power Level
<<parameterized feature>>
12/24 Hour Clock
Recipe
mutually includes

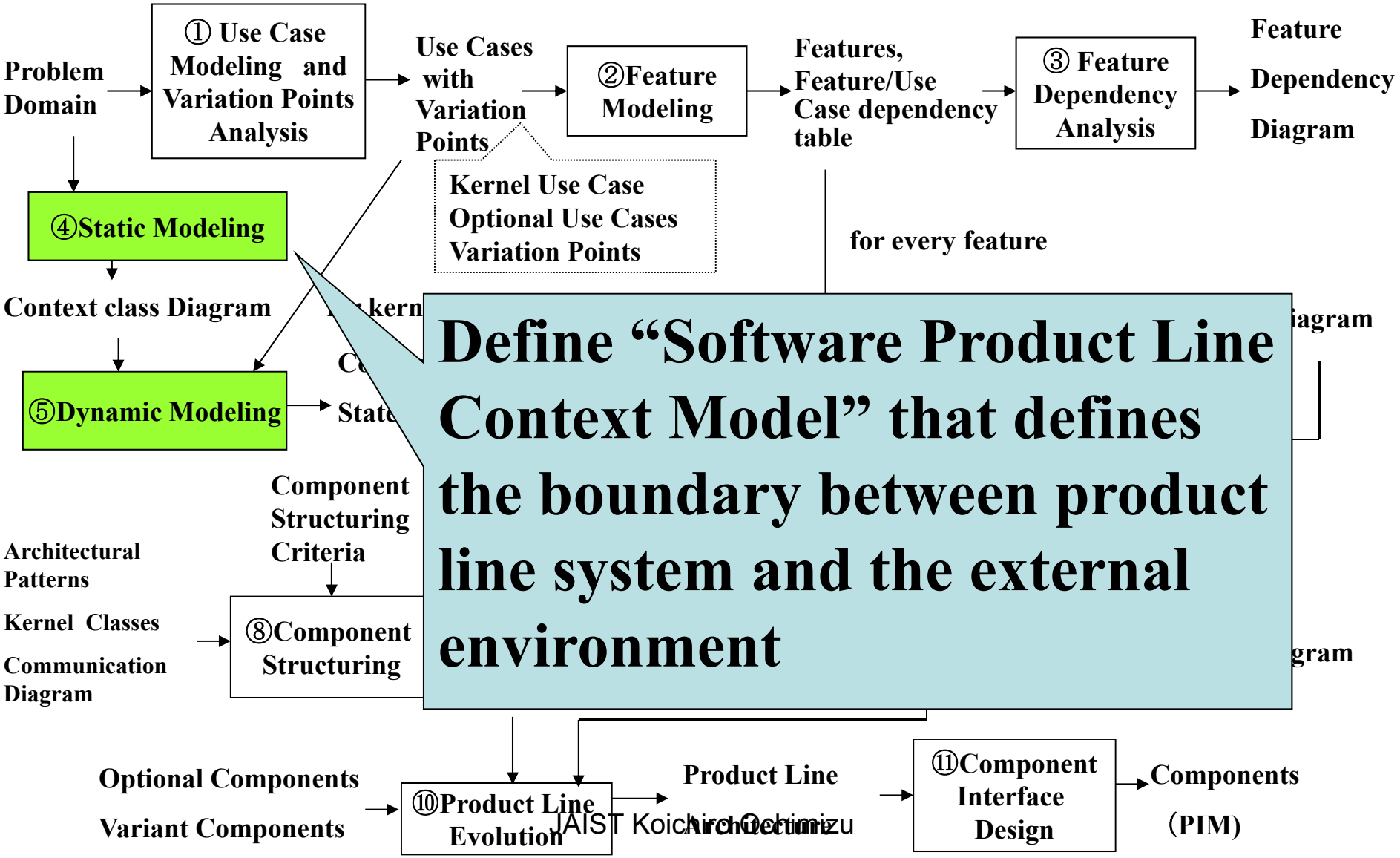

Optional
<<default feature>>
English
<<alternative feature>>
French
<<alternative feature>>
Spanish
<<alternative feature>>
German
<<alternative feature>>
Italian
{mutually exclusive feature}
<<exactly-one-of-feature group>>
Display Language
<<optional feature>>
Minute Plus
<<optional feature>>
Turn Table
<<optional feature>>
Light
<<common feature>>
Microwave Oven Kernel
<<optional feature>>
Beeper
requires
<<exactly-one-of-feature group>>
Display Unit
<<exactly-one-of-feature group>>
Heating Element
<<exactly-one-of-feature group>>
Weight Sensor
{mutually exclusive feature}
<<alternative feature>>
Multi-line Display
<<default feature>>
One-line Display
<<default feature>>
One-level Heating
<<alternative feature>>
Multi-level Heating
<<optional feature>>
TOD Clock
<<parameterized feature>>
12/24 Hour Clock
<<default feature>>
Boolean Weight
<<alternative feature>>
Analog Weigh
<<optional feature>>
Power Level
<<optional feature>>
Recipe
mutually includes


Feature Group
<<default feature>>
English
<<alternative feature>>
French
<<alternative feature>>
Spanish
<<alternative feature>>
German
<<alternative feature>>
Italian
{mutually exclusive feature}
<<exactly-one-of-feature group>>
Display Language
<<optional feature>>
Minute Plus
<<optional feature>>
Turn Table
<<optional feature>>
Light
<<common feature>>
Microwave Oven Kernel
<<optional feature>>
Beeper
requires
<<exactly-one-of-feature group>>
Display Unit
<<exactly-one-of-feature group>>
Heating Element
<<exactly-one-of-feature group>>
Weight Sensor
{mutually exclusive feature}
<<alternative feature>>
Multi-line Display
<<default feature>>
One-line Display
<<default feature>>
One-level Heating
<<alternative feature>>
Multi-level Heating
<<optional feature>>
TOD Clock
<<default feature>>
Boolean Weight
<<alternative feature>>
Analog Weigh
mutually includes
<<parameterized feature>>
12/24 Hour Clock
<<optional feature>>
Power Level
Recipe


Requires
<<default feature>>
English
<<alternative feature>>
French
<<alternative feature>>
Spanish
<<alternative feature>>
German
<<alternative feature>>
Italian
{mutually exclusive feature}
<<exactly-one-of-feature group>>
Display Language
<<optional feature>>
Minute Plus
<<optional feature>>
Turn Table
<<optional feature>>
Light
<<optional feature>>
Beeper
<<common feature>>
Microwave Oven Kernel
requires
<<exactly-one-of-feature group>>
Display Unit
<<exactly-one-of-feature group>>
Heating Element
<<exactly-one-of-feature group>>
Weight Sensor
{mutually exclusive feature}
<<alternative feature>>
Multi-line Display
<<default feature>>
One-line Display
<<default feature>>
One-level Heating
<<alternative feature>>
Multi-level Heating
<<optional feature>>
TOD Clock
<<default feature>>
Boolean Weight
<<alternative feature>>
Analog Weigh
<<parameterized feature>>
12/24 Hour Clock
<<optional feature>>
Power Level
mutually includes
JAIST Kokichi Sugihara
Recipe

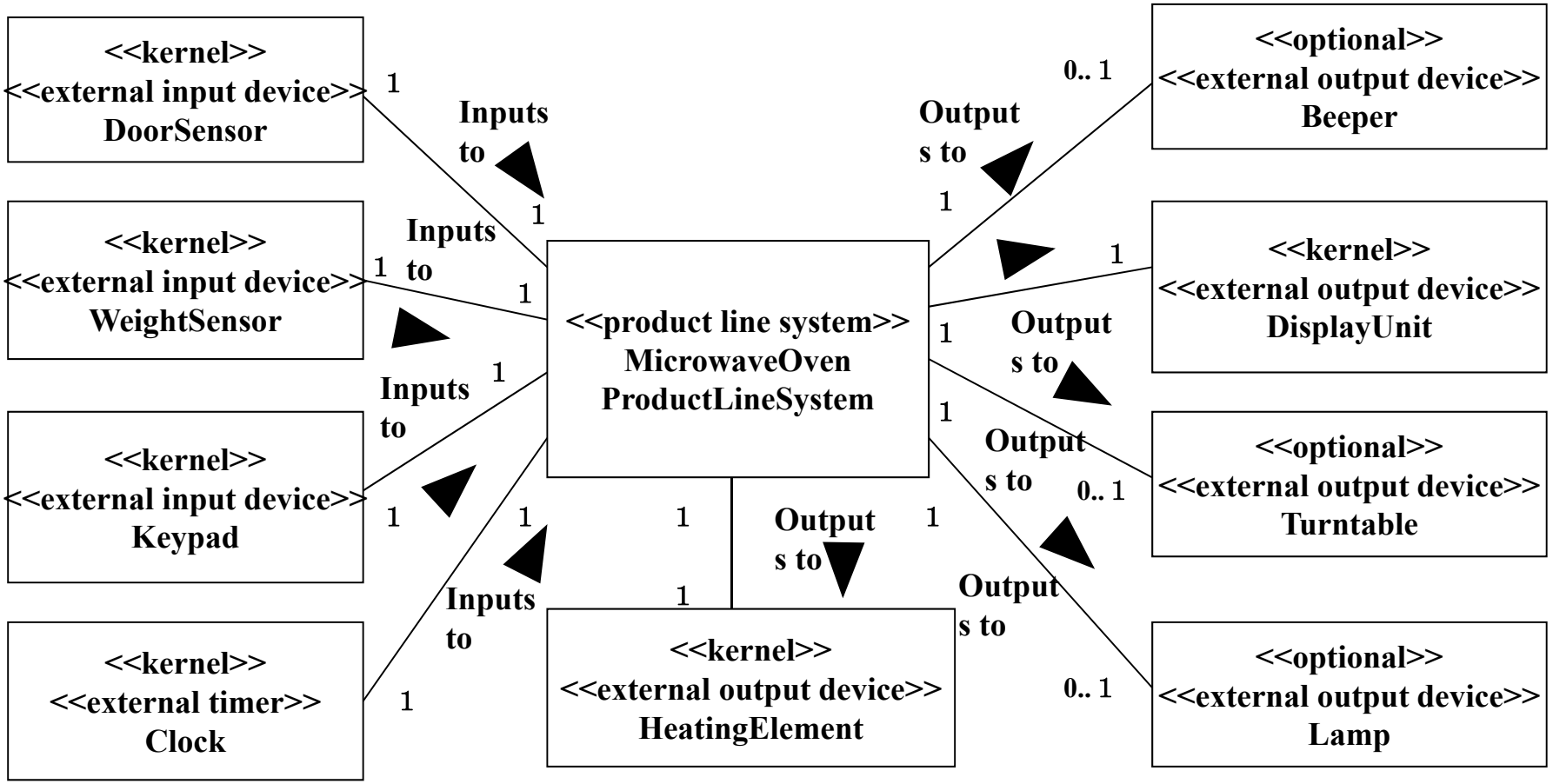

mutually includes
<<default feature>>
English
<<alternative feature>>
French
<<alternative feature>>
Spanish
<<alternative feature>>
German
<<alternative feature>>
Italian
{mutually exclusive feature}
<<exactly-one-of-feature group>>
Display Language
<<optional feature>>
Minute Plus
<<optional feature>>
Turn Table
<<optional feature>>
Light
<<optional feature>>
Beeper
<<common feature>>
Microwave Oven Kernel
requires
<<exactly-one-of-feature group>>
Display Unit
<<exactly-one-of-feature group>>
Heating Element
<<exactly-one-of-feature group>>
Weight Sensor
{mutually exclusive feature}
<<alternative feature>>
Multi-line Display
<<default feature>>
One-line Display
<<default feature>>
One-level Heating
<<alternative feature>>
Multi-level Heating
<<optional feature>>
TOD Clock
<<default feature>>
Boolean Weight
<<alternative feature>>
Analog Weigh
<<parameterized feature>>
12/24 Hour Clock
<<optional feature>>
Power Level
mutually includes
Recipe

Outline of PLUS (Finding problem domain objects)



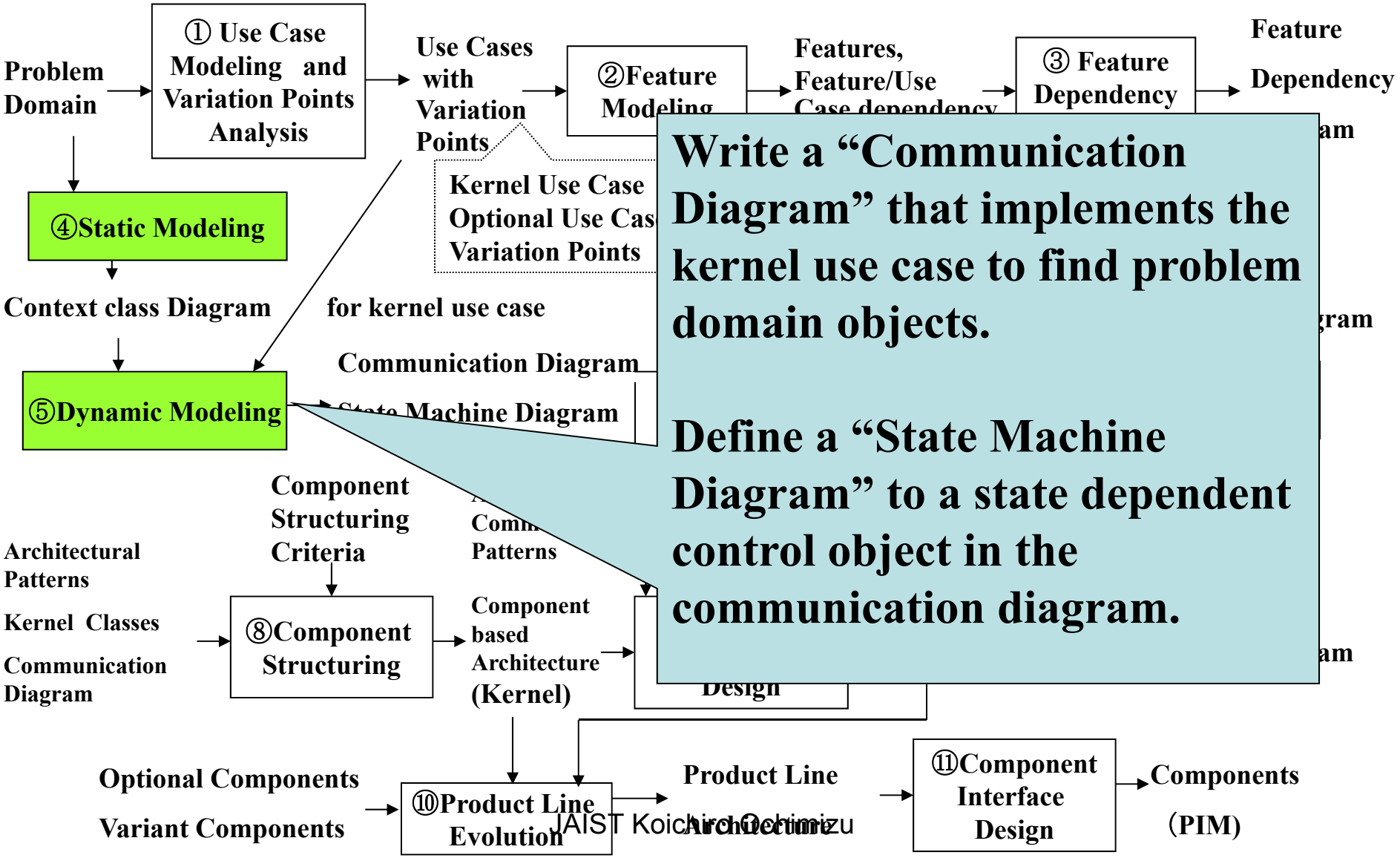
Software Product Line Context Model

- The product line context class diagram defines the boundary between a product line system (i.e. any member of the product line) and the external environment(i.e. the external classes to which members of the product line have to interface)



Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Outline of PLUS (Finding problem domain objects)

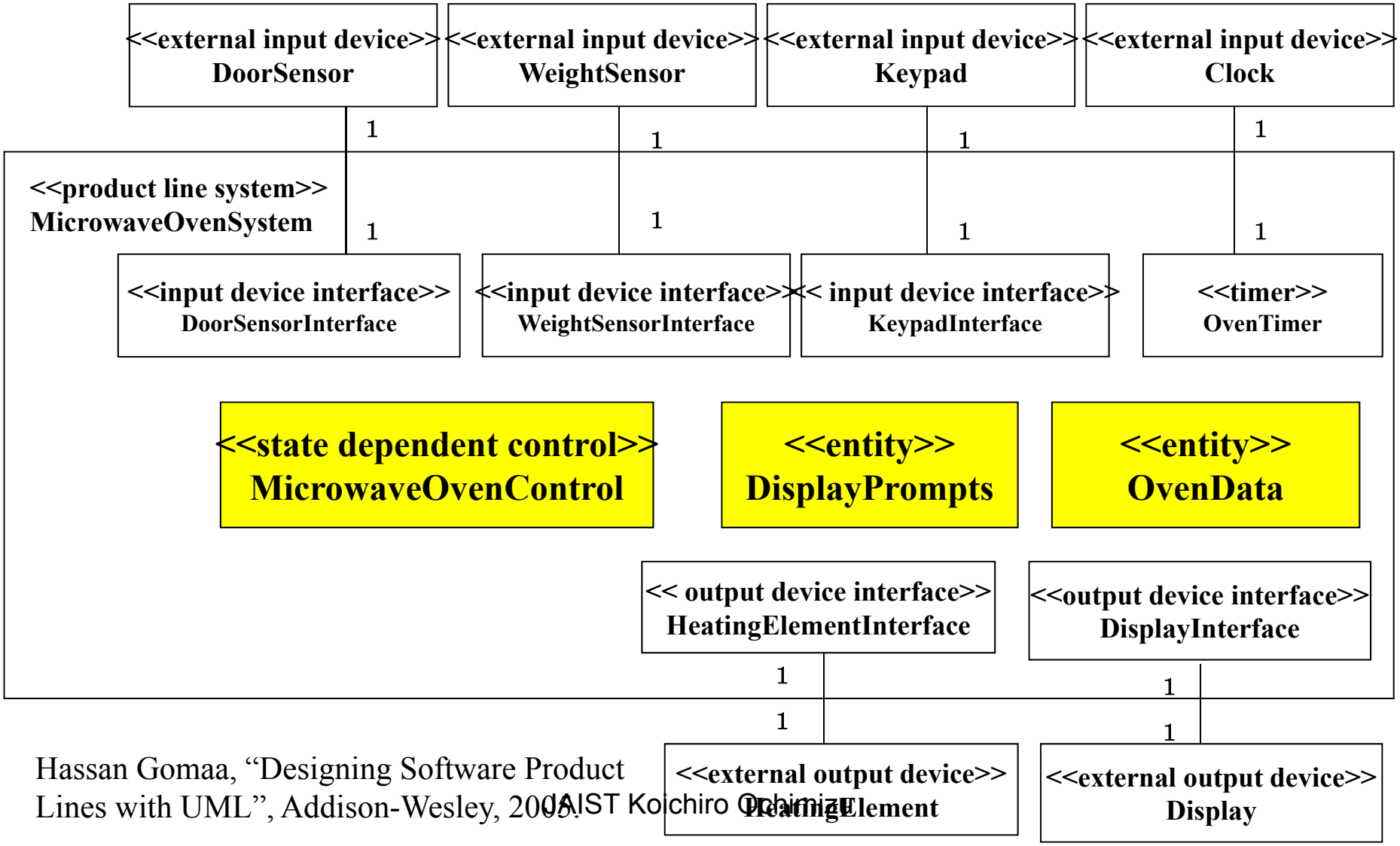


Dynamic Modeling

- Develop a dynamic model of the product line after the external classes have been determined in the system context model.
- The kernel of the product line is analyzed first with the forward evolutionary engineering strategy.
  - Initially the kernel classes are determined by consideration of the kernel use cases and the interaction among those objects on communication diagrams and state machines.
  - After that, the optional and variant classes are determined by consideration of variation points (as determined from the use case and feature models) and the optional use cases.

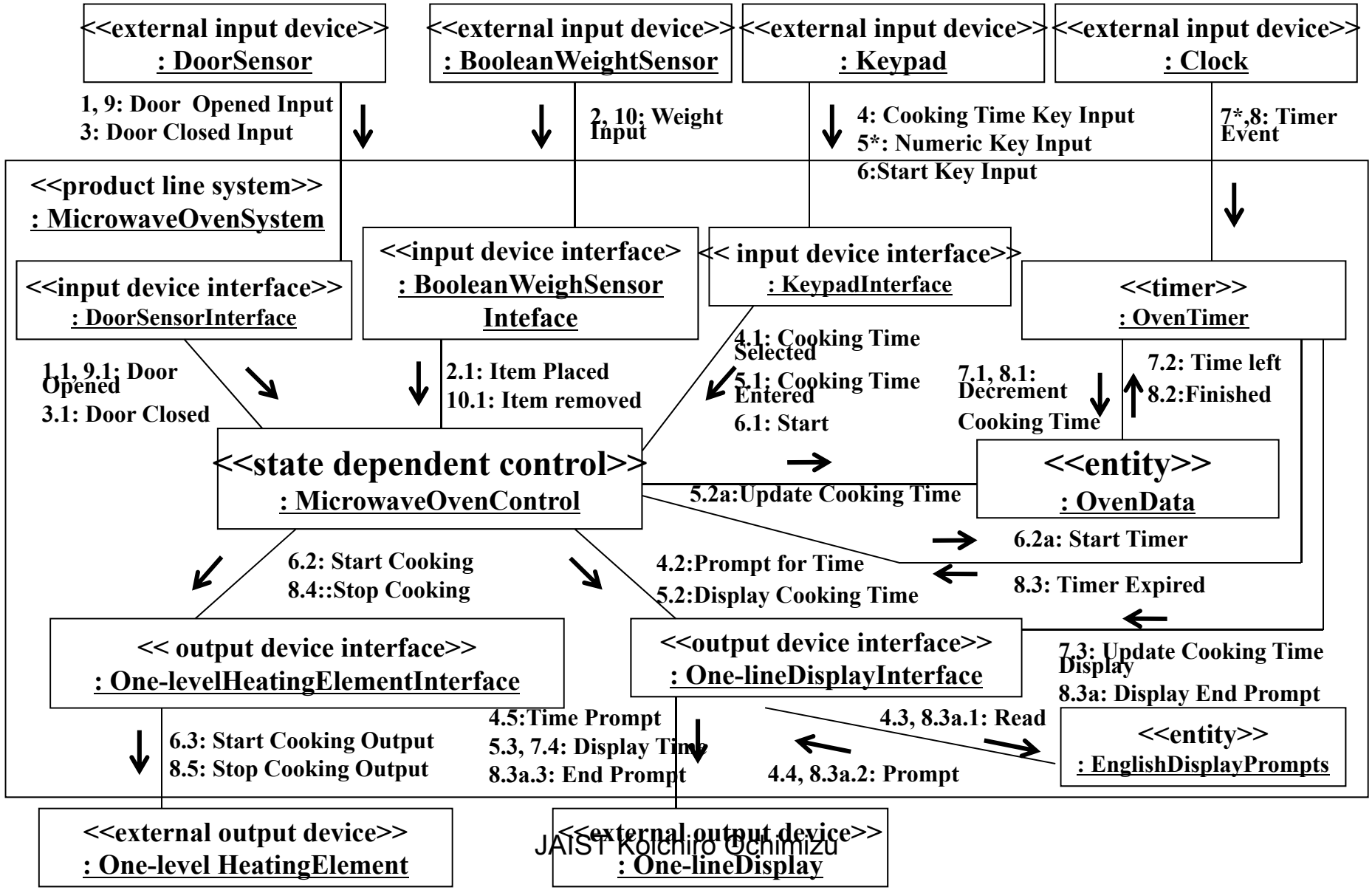
Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Kernel classes



Hassan Gomaa, "Designing Software Product Lines with UML", Addison-Wesley, 2005.

Communication diagram for kernel use case “Cook Food”



The Sequence of messages for the kernel communication Diagram(1/7)

1. **Door Opened Input.** The user opens the door. The external Door Sensor object sends this input to the Door Sensor Interface object.

1.1 Door Opened. Door Sensor Interface sends the Door Opened message to the Microwave Oven Control object , which changes state.

2. Weight Input. The user places an item to be cooked into the oven. The external Boolean Weight Sensor object sends this input to the Boolean Weight Sensor Interface object.

2.1 Item Placed. Boolean Weight Sensor Interface sends the Item Placed message to the Microwave Oven Control object, which changes state.

3. Door Closed Input. The user closes the door. The external Door Sensor object sends this input to the Door Sensor Interface object.

3.1 Door Closed. Door Sensor Interface sends the Door Closed message to the Microwave Oven Control object, which changes state.

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

The Sequence of messages for the kernel communication diagram(2/7)

4. Cooking Time Key Input. The user presses the **Cooking Time** button on the keypad. The external Keypad object sends this input to the Keypad Interface object.

4.1 Cooking Time Selected. Keypad Interface sends the Cooking Time Selected message to the Microwave Oven Control object, which changes state.

4.2 Prompt for Time. As a result of changing state, Microwave Oven Control object sends Prompt for Time message to the One-line Display Interface object.

4.3 Read. The message arriving at One-line Display Interface contains a prompt ID, so One-line Display Interface sends a Read message to English Display Prompts to get the corresponding prompt message.

4.4 Prompt. English Display Prompts returns the text for the Time Prompt message.

4.5 Time Prompt. One-line Display Interface sends the Time Prompt output to the external One-line Display object.

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

The Sequence of messages for the kernel communication diagram(3/7)

5* Numeric Key Input. The user enters the numeric value of the time on the keypad, pushing one or more keys. Keypad sends the value of the numeric key(s) input to Keypad Interface.

5.1 Cooking Time Entered. Keypad Interface sends the internal value of each numeric key to Microwave Oven Control.

5.2 Display Cooking Time. Microwave Oven Control sends the value of each numeric key to One-line Display Interface, to ensure that these values are sent only in the appropriate state.

5.2a Update Cooking Time. Microwave Oven Control concurrently sends the numeric value of each numeric key to Oven Data to Update the cooking time.

5.3 Display Time. One-line Display Interface shifts the previous digit to the left and adds the new digit. It then sends the new value of cooking time to the external One-line Display object.

The Sequence of messages for the kernel communication diagram(4/7)

6. Start Key Input. The user presses the **Start** button. The external Keypad object sends his input to the Keypad Interface object.

6.1 Start. Keypad Interface sends the Start message to Microwave Oven Control, which changes state.

6.2 Start Cooking. As a result of changing state, Microwave Oven Control sends the Start Cooking message to the One-level Heating Element Interface object.

6.2a Start Timer. Microwave Oven Control concurrently notifies the Oven Timer to start the oven timer.

6.3 Start Cooking Output. One-level Heating Element Interface sends this output to One-level Heating Element to start cooking the food.

Hassan Gomaa, "Designing Software Product Lines with UML", Addison-Wesley, 2005.

The Sequence of messages for the kernel communication diagram(5/7)

7* Timer Event. The external Clock object sends a timer event every second to Oven timer.

7.1 Decrement Cooking Time. As Oven Timer is counting, I sends this message to the Oven Data object, which maintains the cooking time.

7.2 Time Left. After decrementing the cooking time, which is assumed to be greater than zero at this step of the scenario, Oven Data sends the Time Left message to Oven Timer.

7.3 Update Cooking Time Display. Oven Timer sends the cooking time left to One-line Display Interface.

7.4 Display Time. One-line Display Interface outputs the new cooking time value to the external One-line Display object.

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

The Sequence of messages for the kernel communication diagram(6/7)

8 Timer Event. The external Clock object sends a timer event every second to Oven Timer

8.1 Decrement Cooking Time. As Oven Timer is counting, it sends this message to the Oven Data object, which maintains the cooking time.

8.2 Finished. After decrementing the cooking time, which is assumed to be equal to zero at this step of the scenario, Oven Data sends the Finished message to Oven Timer.

8.3 Timer Expired. Oven Timer sends the Timer Expired message to Microwave Oven Control, which changes state.

8.3a Display End Prompt. Oven Timer concurrently sends the Display End Prompt message to One-line Display Interface.

8.3a.1 Read. The message arriving at One-line Display Interface contains a prompt ID, so One-line Display Interface sends a Read message to English Display Prompts to get the corresponding prompt message.

8.3a.2 Prompt. English Display Prompts returns the text for the End Prompt message,

8.3a.3 End Prompt. One-line Display Interface outputs the End Prompt message to the external One-line Display object.

8.4 Stop Cooking. As a result of changing state(in step 8.3), Microwave Oven Control sends the Stop Cooking message to One-level Heating Element Interface object.

8.5 Stop Cooking Output. One-level Heating Element Interface sends this output to the One-level Heating Element object to stop cooking the food.

The Sequence of messages for the kernel communication diagram(7/7)

9 Door Opened Input. The user opens the door. The external Door Sensor object sends this input to the Door Sensor Interface object.

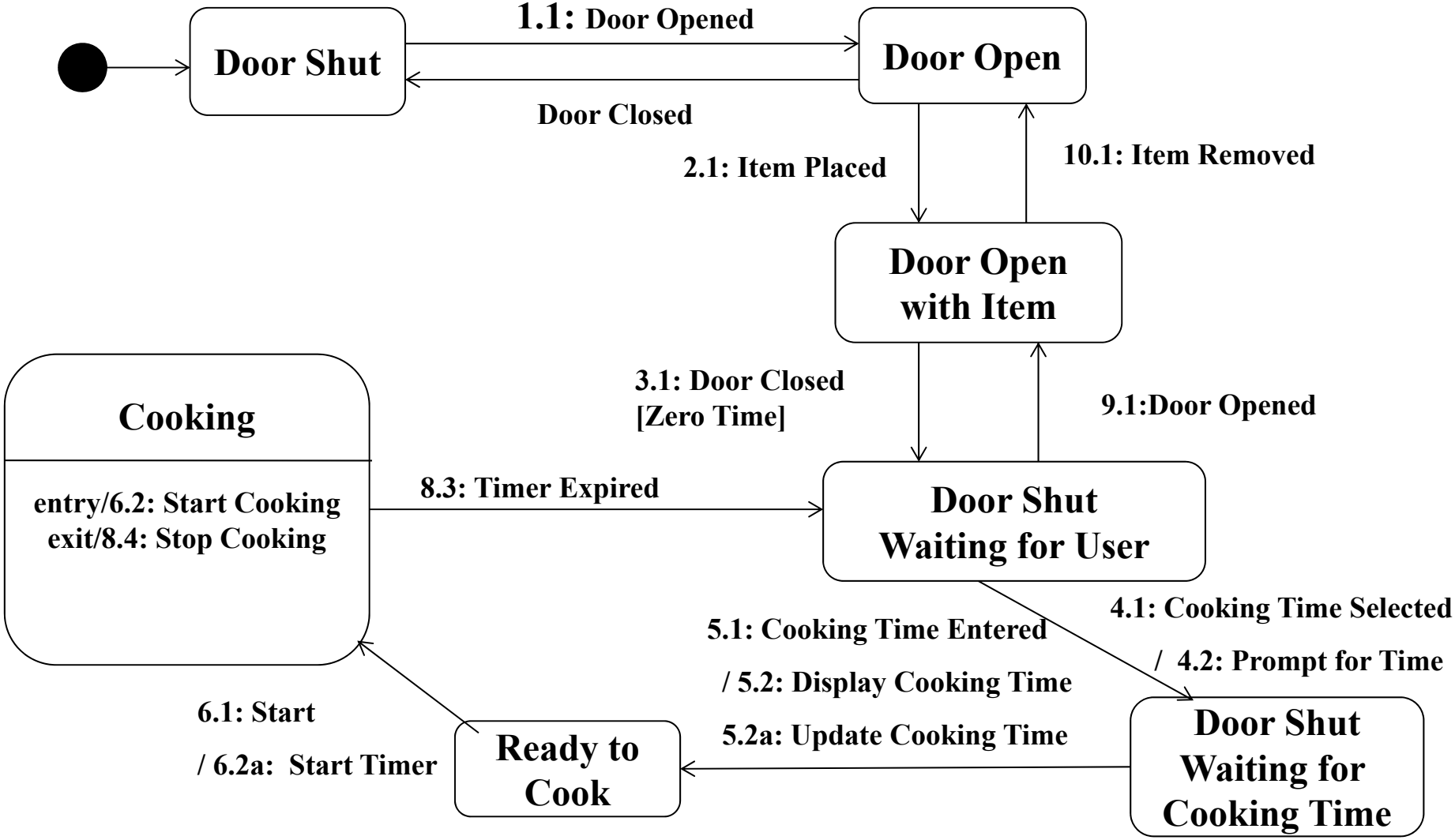
9.1 Door Opened. Door Sensor Interface sends the Door Opened message to the Microwave Oven Control object, which changes state.

10 Weight Input. The user removes the cooked item from the oven. The external Boolean Weight Sensor object sends this input to the Boolean Weight Sensor Interface object.

10.1 Item Removed. Boolean Weight Sensor Interface sends the Item Removed message to the Microwave Oven Control object, which changes state.

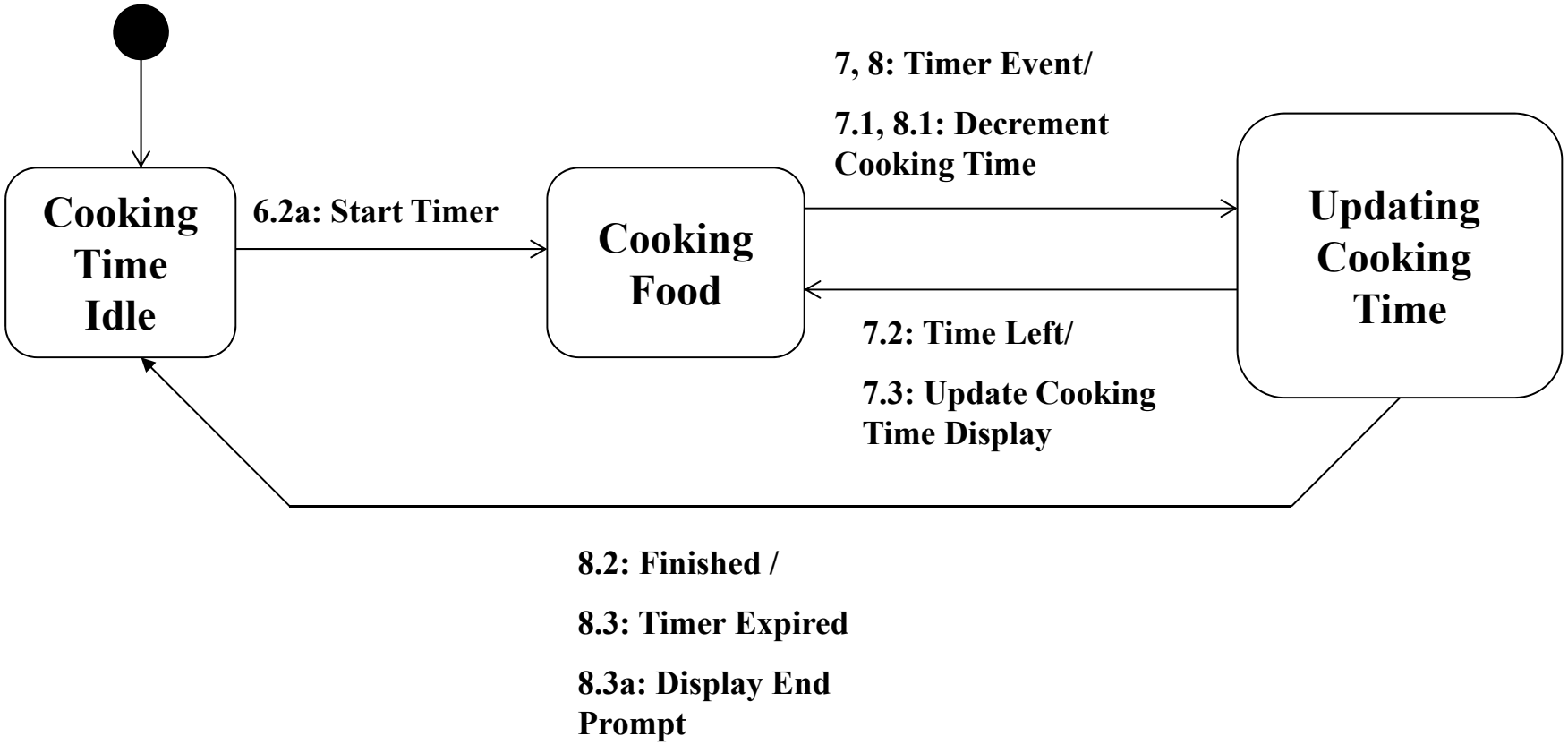
Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

State Machine for Micro Oven Control : for one scenario representing the main sequence through Use Case "Cook Food"



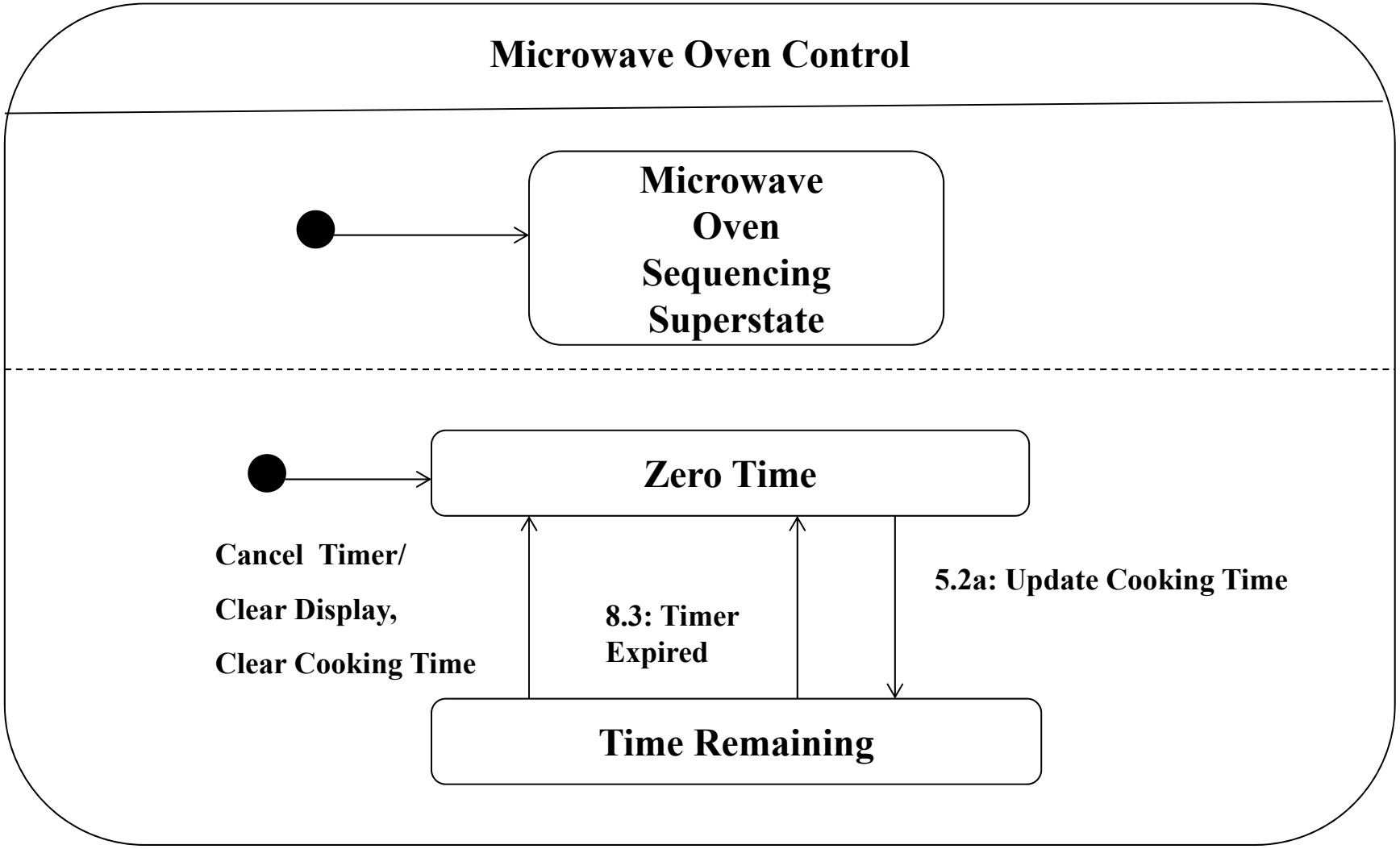
Hassan Gomaa, "Designing Software Product Lines with UML", Addison-Wesley, 2005.

State Machine for Oven Timer



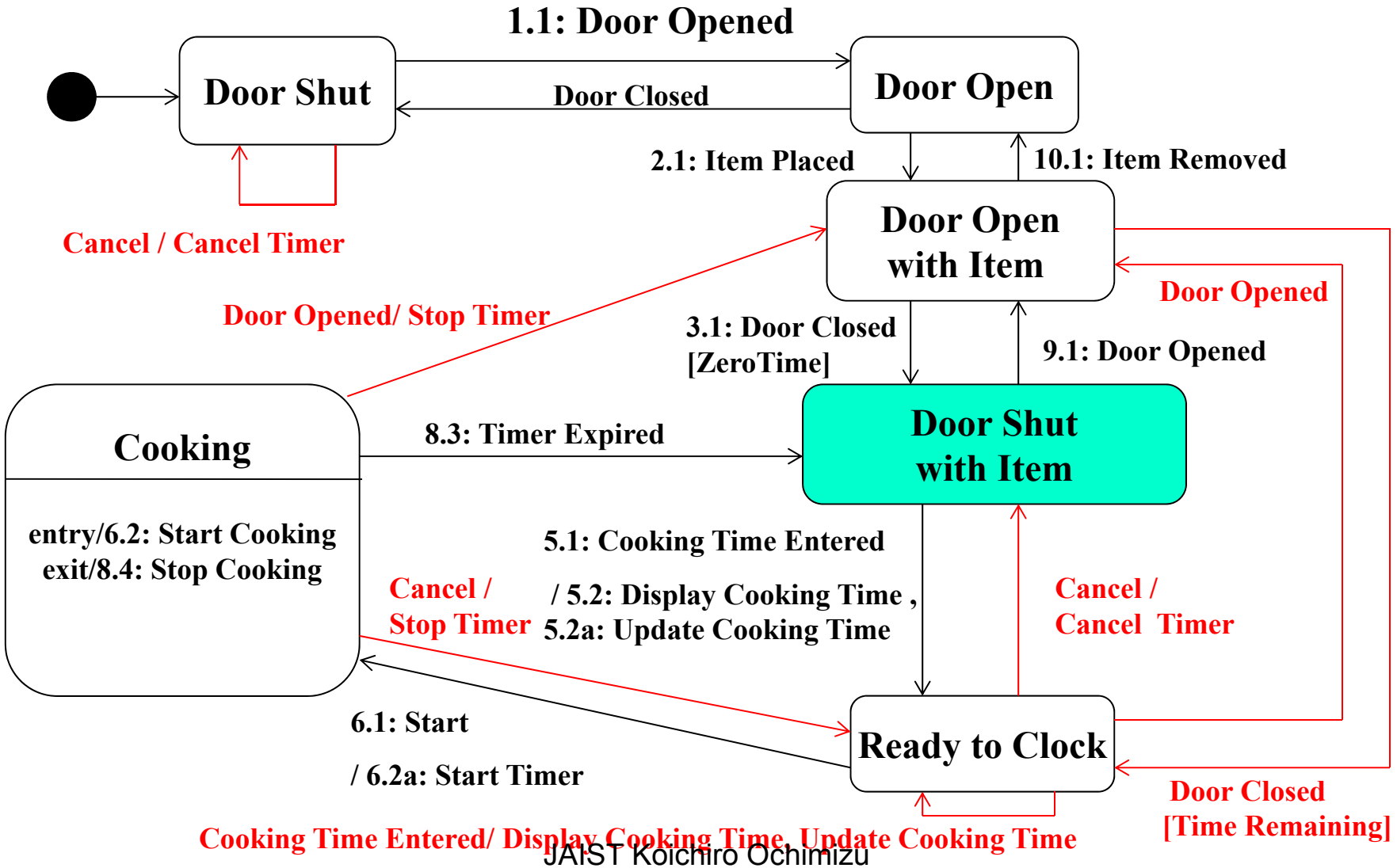
Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

State Machine for Microwave Oven Control (Kernel top-level State Machine)



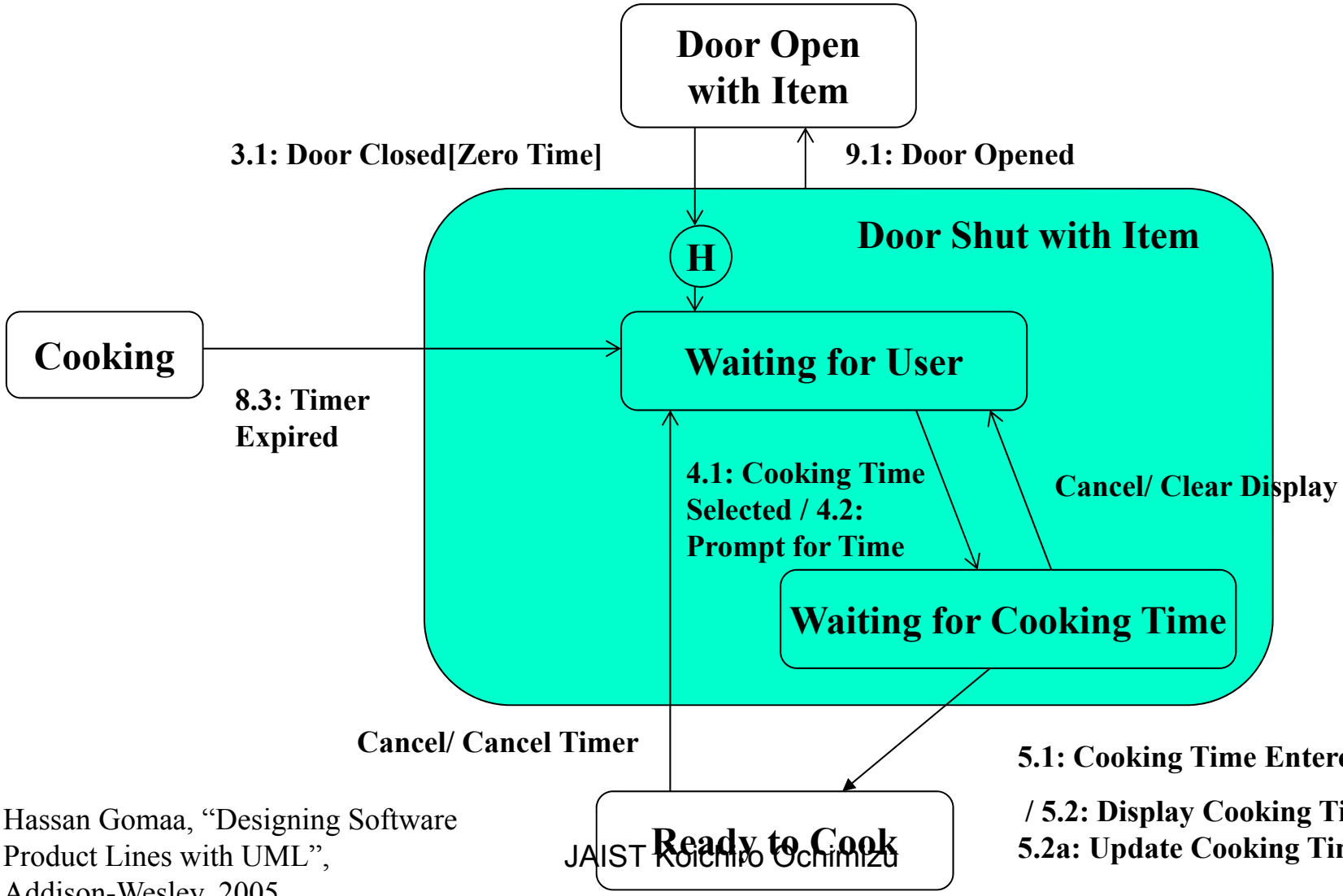
Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Decomposition of the Microwave Oven Sequencing Superstate



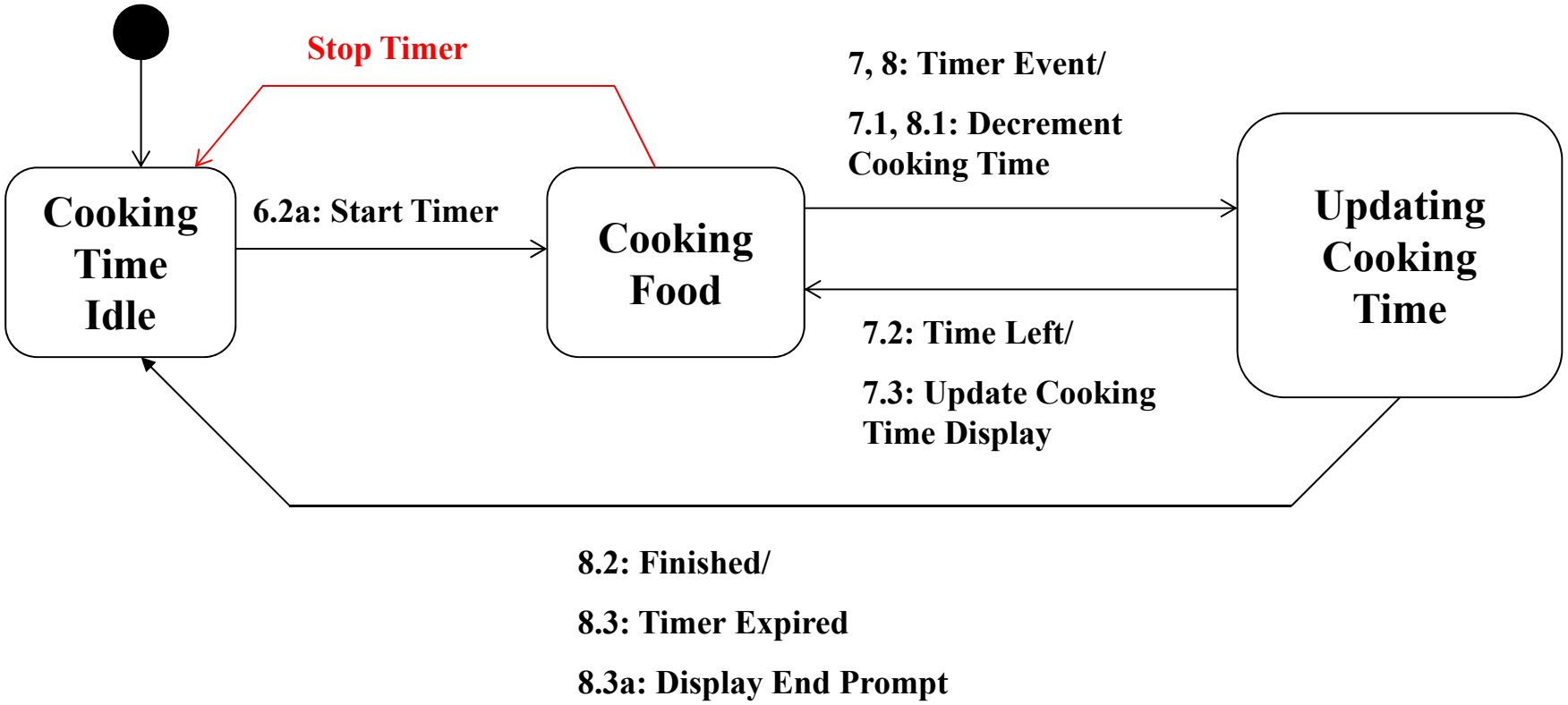
Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Decomposition of the superstate “Door Shut with Item”



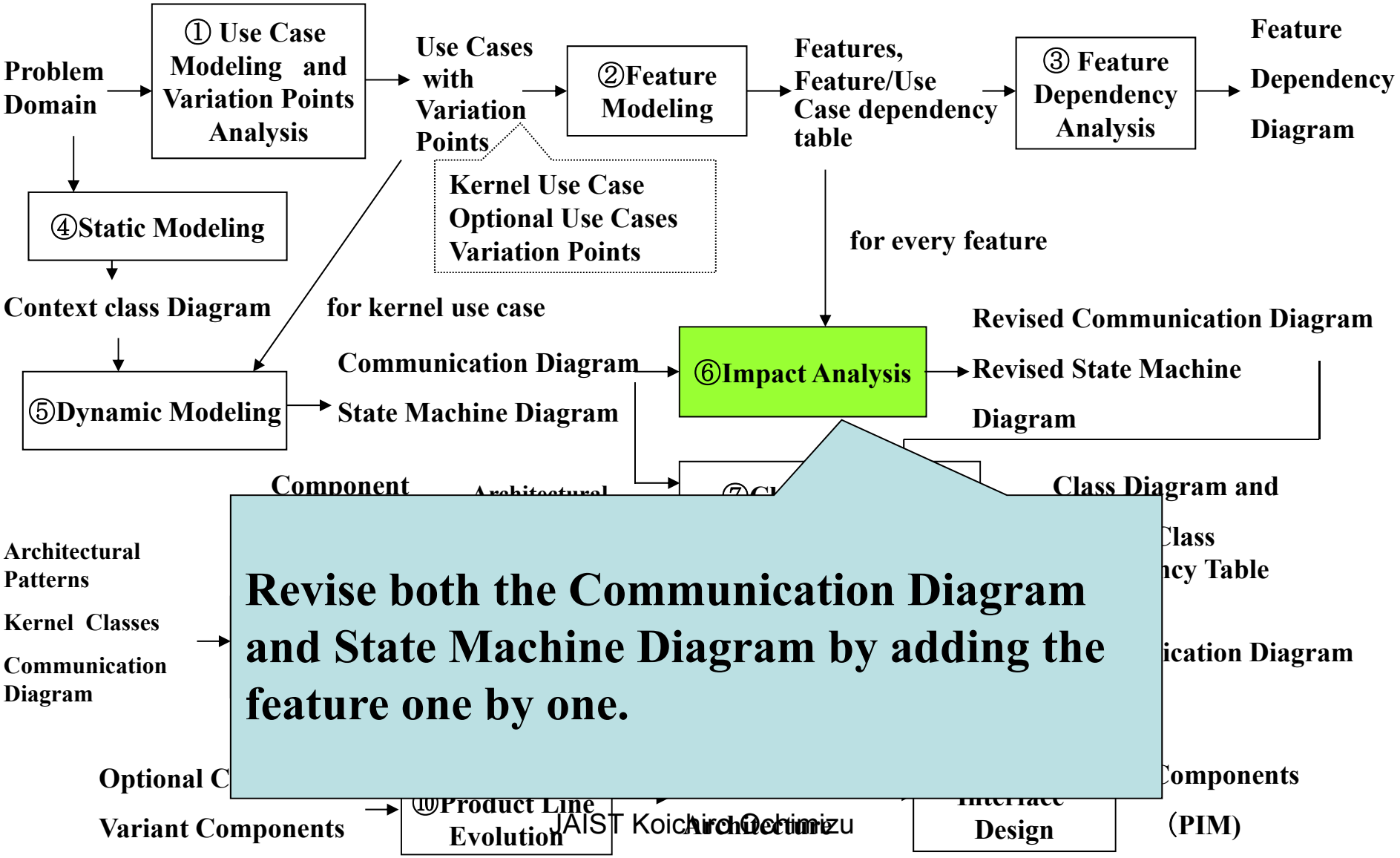
Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Kernel State Machine for Oven Timer



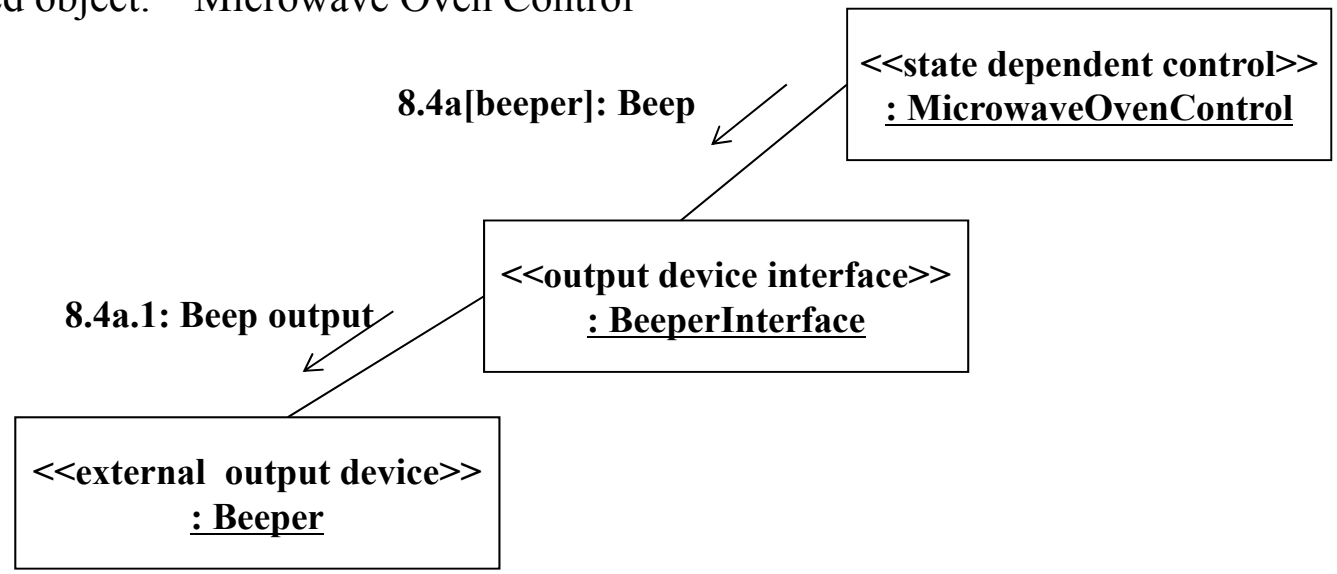
Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Outline of PLUS (Impact Analysis)



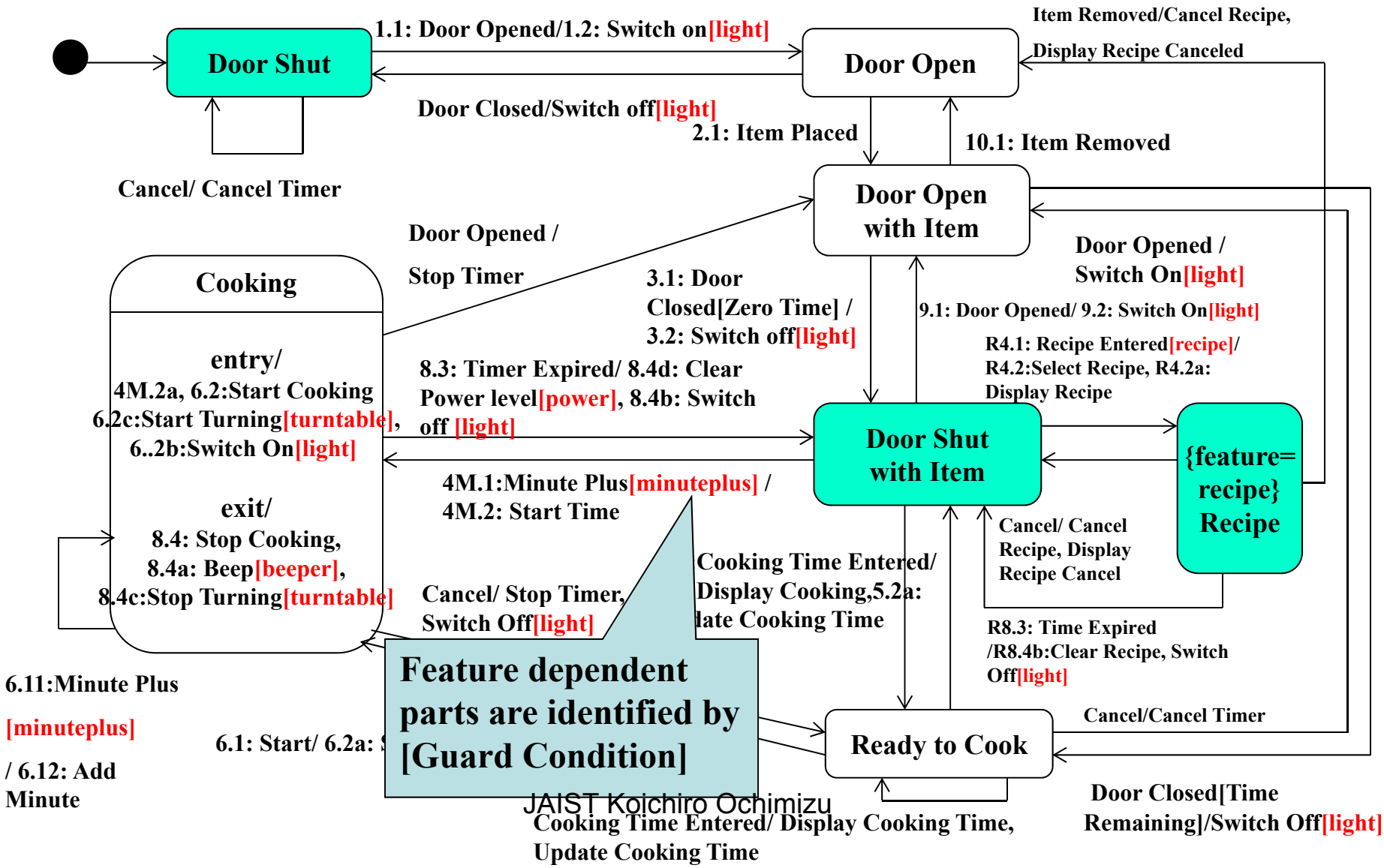
Impact Analysis of the Beeper Feature

- The beeper is Switched on when cooking has finished
- Impact 1: The need for the Beeper external output device and the Beeper Interface output device interface object.
- Impact 2: The Microwave Oven Control object is the state-dependent control object that sends the Beep command to Beeper Interface when cooking is stopped. The impact on the Microwave Oven Control state machine is that it needs to have an optional Beep action, which is also guarded by the [beeper] feature condition
- <<optional feature>> Beeper
  optional object: Beeper Interface
  affected object: Microwave Oven Control

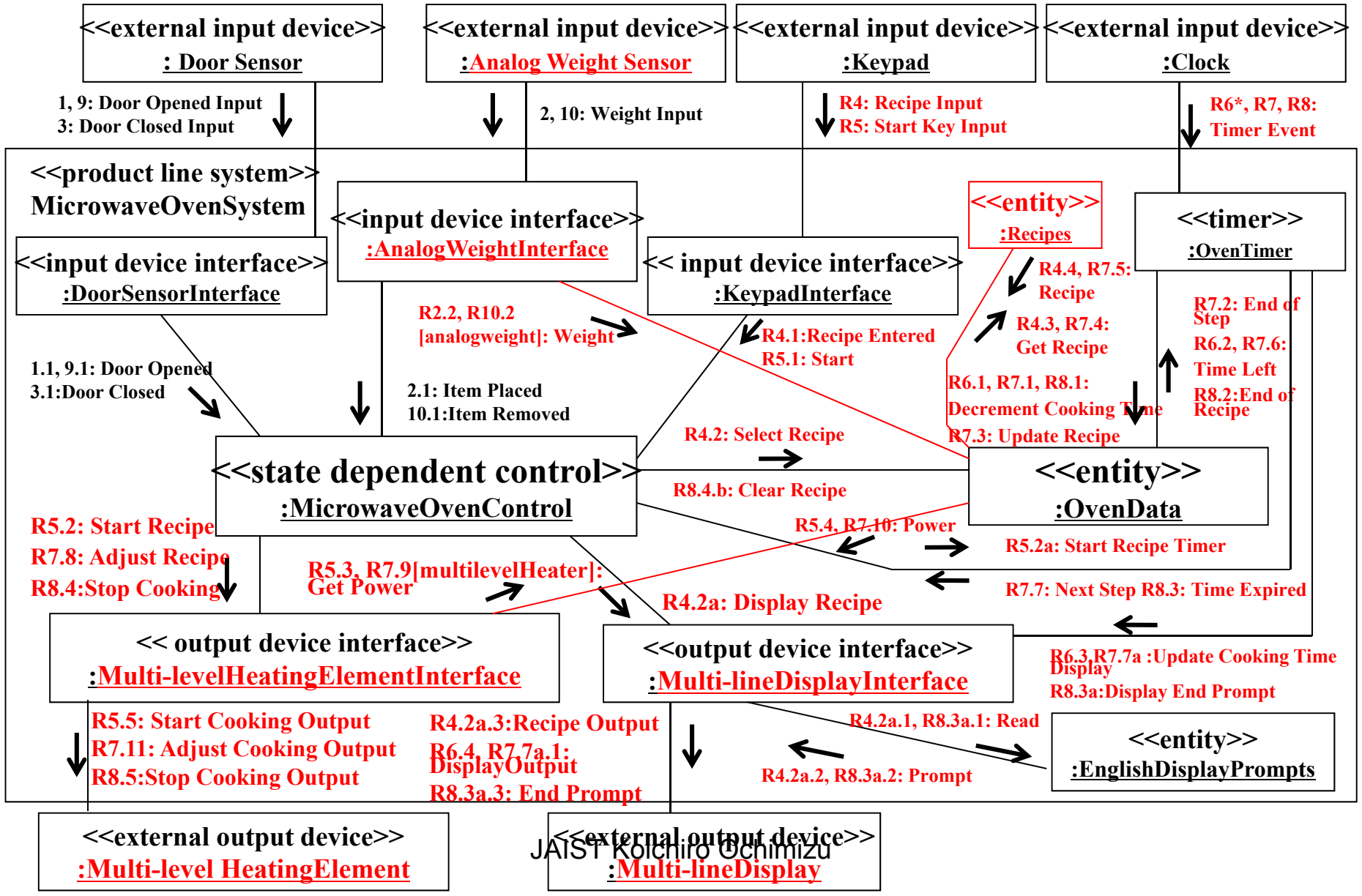


Hassan Gomaa, "Designing Software Product Lines with UML", Addison-Wesley, 2005.

State Machine for Microwave Control with all features

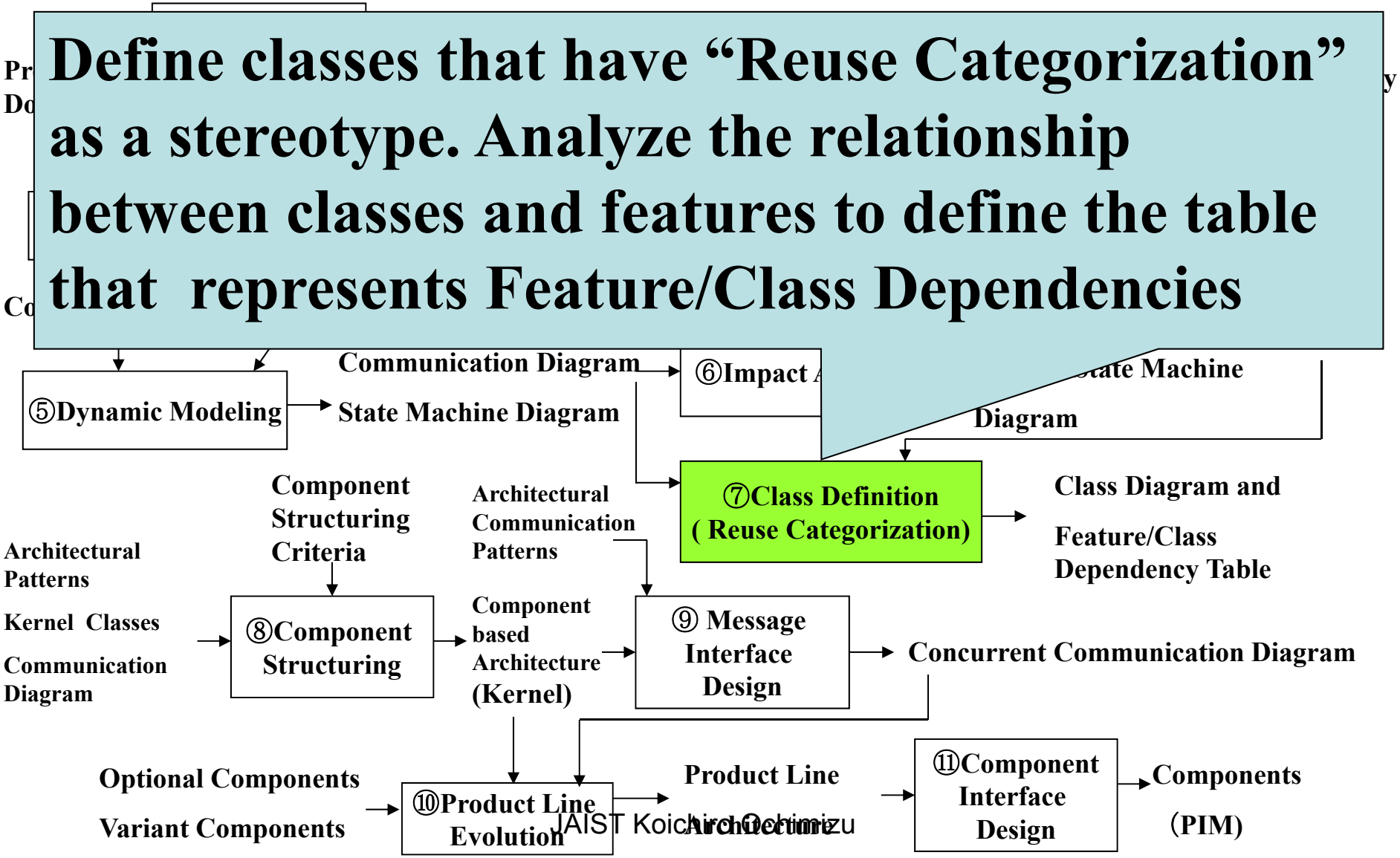


Result of Impact analysis for Recipe feature !



Outline of PLUS (Feature/Class Dependency Analysis)

Define classes that have "Reuse Categorization" as a stereotype. Analyze the relationship between classes and features to define the table that represents Feature/Class Dependencies



Feature/Class Dependency Analysis

- The Impact analyses identifies both the optional objects and the affected objects.
- Optional objects are new objects that were not used in the kernel communication diagrams but are needed to support an optional or alternative feature.
- Affected objects are objects that must behave differently to support an optional or alternative feature.
- For the affected objects, an important decision is whether to handle the change by using inheritance or by parameterization.
- Each of the classes is considered from a product line reuse perspective.

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Class Reuse Categorization for Software Product line

- Main class reuse category
  - <<**kernel**>> A class provided by every member of the product line and used without change by every member.
  - <<**optional**>> A class provided by some members of the product line but not all. When used, it is used without change.
  - <<**variant**>> One of a set of similar classes, which have some identical properties but others that are different. Different variant classes are used by different members of the product line.
  - <<**default**>> The default class among a set of variant classes, which is provided by some members of the product line.

Hassan Gomaa, "Designing Software Product Lines with UML", Addison-Wesley, 2005.

Parameterized Class Reuse Categories

- Parameterized class reuse category
  - <<kernel-param-vp>> Kernel. The values of the configuration parameters need to be set by the individual product line member. Example: Microwave Oven Control
  - <<optional-param-vp>> Optional. The values of the configuration parameters need to be set by the individual product line member. Example: TOD Timer
  - <<variant-param-vp>> Variant. The values of the configuration parameters need to be set by the individual product line member.
  - <<default-param-vp>> Default. The values of the configuration parameters need to be set by the individual product line member.

Hassan Gomaa, "Designing Software Product Lines with UML", Addison-Wesley, 2005.

Abstract/Concrete Class Reuse Categories

- abstract class reuse category
  - <<kernel-abstract-vp>> An abstract class provided by every member of the product line.
  - <<optional-abstract-vp>> An abstract class provided by some members of the product line but not all.
- concrete class reuse category
  - <<kernel-vp>> Concrete sub class of kernel-abstract-vp
  - <<optional-vp>> Concrete sub class of optional-abstract-vp
  - <<variant-vp>> One of a set of concrete variant classes.

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Feature/class Dependencies(1/6)

Feature Name	Feature Category	Class Name	Class Category	Class Parameter
Microwave Oven Kernel	common	Door Sensor Interface	kernel	
		Weight Sensor Interface	kernel-abstract-vp	
		Keypad Interface	kernel-param-vp	
		Heating Element Interface	kernel-abstract-vp	
		Display Interface	kernel-abstract-vp	
		Microwave Oven Control	kernel-param-vp	
		Oven Timer	kernel-param-vp	
		Oven Data	kernel-param-vp	
		Display Prompts	kernel-abstract-vp	

Feature/class Dependencies(2/6)

Feature Name	Feature Category	Class Name	Class Category	Class Parameter
Light	optional	Lamp Interface	optional	
		Microwave Oven Control	kernel-param-vp	light:Boolean
Turntable	optional	Turntable Interface	optional	
			kernel-param-vp	turntable:Boolean
Beeper	optional	Beeper Interface	optional	
		Microwave Oven Control	kernel-param-vp	beeper:Boolean
Minute Plus	optional	Keypad Interface	kernel-param-vp	minuteplus:Boolean
		Microwave Oven Control	kernel-param-vp	minuteplus:Boolean
		Oven Timer	kernel-param-vp	minuteplus:Boolean
		Oven Data	kernel-param-vp	minuteplus:Boolean

Feature/class Dependencies(3/6)

Feature Name	Feature Category	Class Name	Class Category	Class Parameter
One-line Display	default	One-line Display Interface	default	
Multi-line Display	alternative	Multi-line Display Interface	variant	
English	default	English Display Prompts	default	
French	alternative	French Display Prompt	variant	
Spanish	alternative	Spanish Display Prompts	variant	
German	alternative	German Display Prompts	variant	
Italian	alternative	Italian Display Prompts	variant	

Feature/class Dependencies(4/6)

Feature Name	Feature Category	Class Name	Class Category	Class Parameter
Boolean Weight	default	Boolean Weight Sensor Interface	default	
Analog Weight	alternative	Analog Weight Sensor	variant	
		Oven Data	kernel-param-vp	itemWeight:Real
One-level Heating	default	One-level Heating Element Interface	default	
Multi-level Heating	alternative	Multi-level Heating Element Interface	variant	
		Microwave Oven Control	kernel-param-vp	Multi-levelHeater: Boolean
		Oven Data	kernel-param-vp	selectedPowerLevel: Integer
Power Level	optional	Keypad Interface	kernel-param-vp	power: Boolean
		Microwave Oven Control	kernel-param-vp	power: Boolean

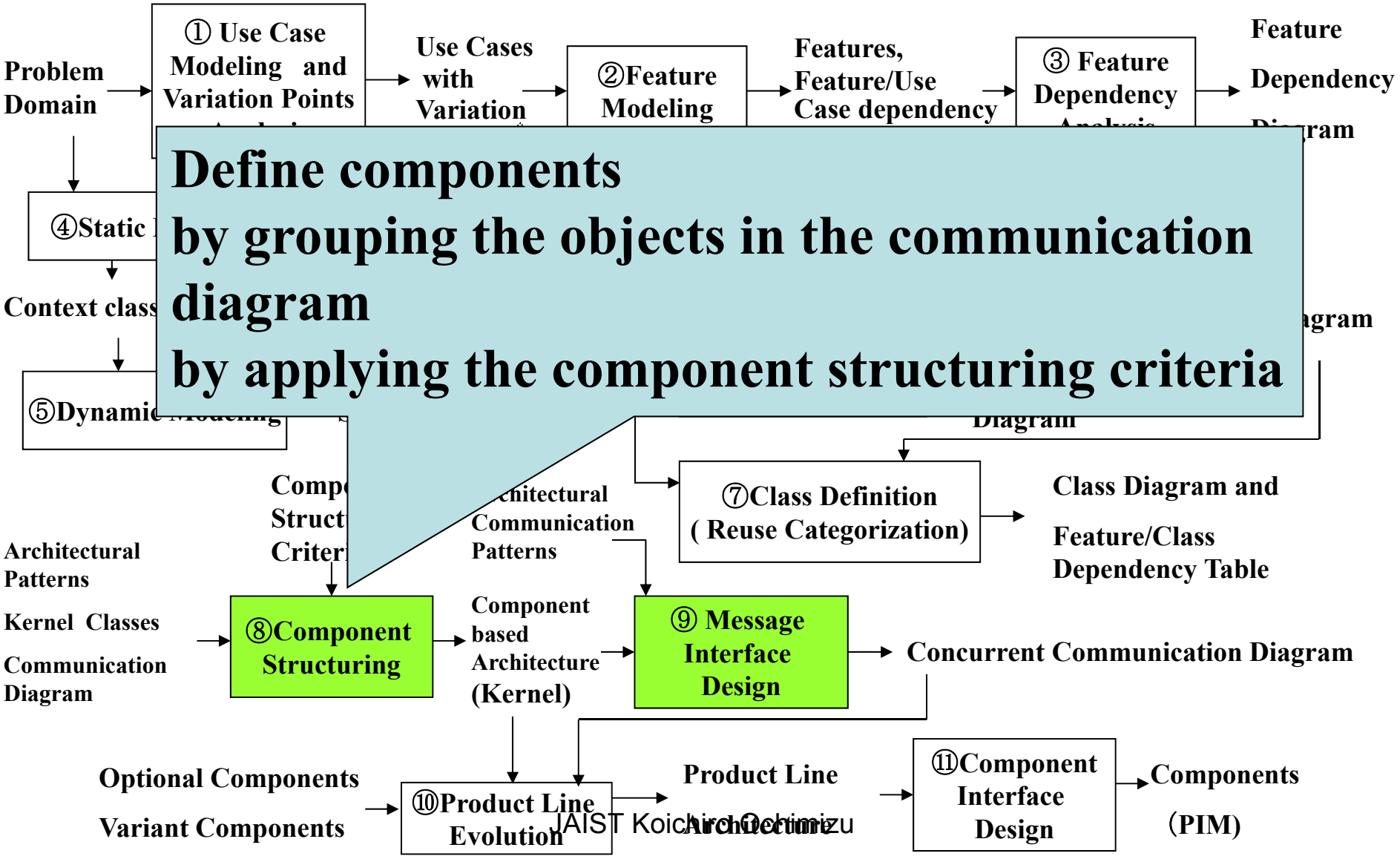
Feature/class Dependencies(5/6)

Feature Name	Feature Category	Class Name	Class Category	Class Parameter
TOD Clock	optional	TOD Timer	optional	
		Keypad Interface	kernel-param-vp	TODClock: Boolean
		Microwave Oven Control	kernel-param-vp	TODClock: Boolean
		Oven Data	kernel-param-vp	TODvalue: Real
12/24 Hour Clock	parameterized	Oven Data	kernel-param-vp	TODmaxHour: Integer

Feature/class Dependencies(6/6)

Feature Name	Feature Category	Class Name	Class Category	Class Parameter
Recipe	optional	Recipes	optional	
		Recipe	optional	
		Keypad Interface	kernel-param-vp	recipe: Boolean
		Microwave Oven Control	kernel-param-vp	recipe: Boolean
		Oven Data	kernel-param-vp	selectedRecipe: Integer
		Oven Timer	kernel-param-vp	recipe: Boolean

Outline of PLUS (Component Structuring)

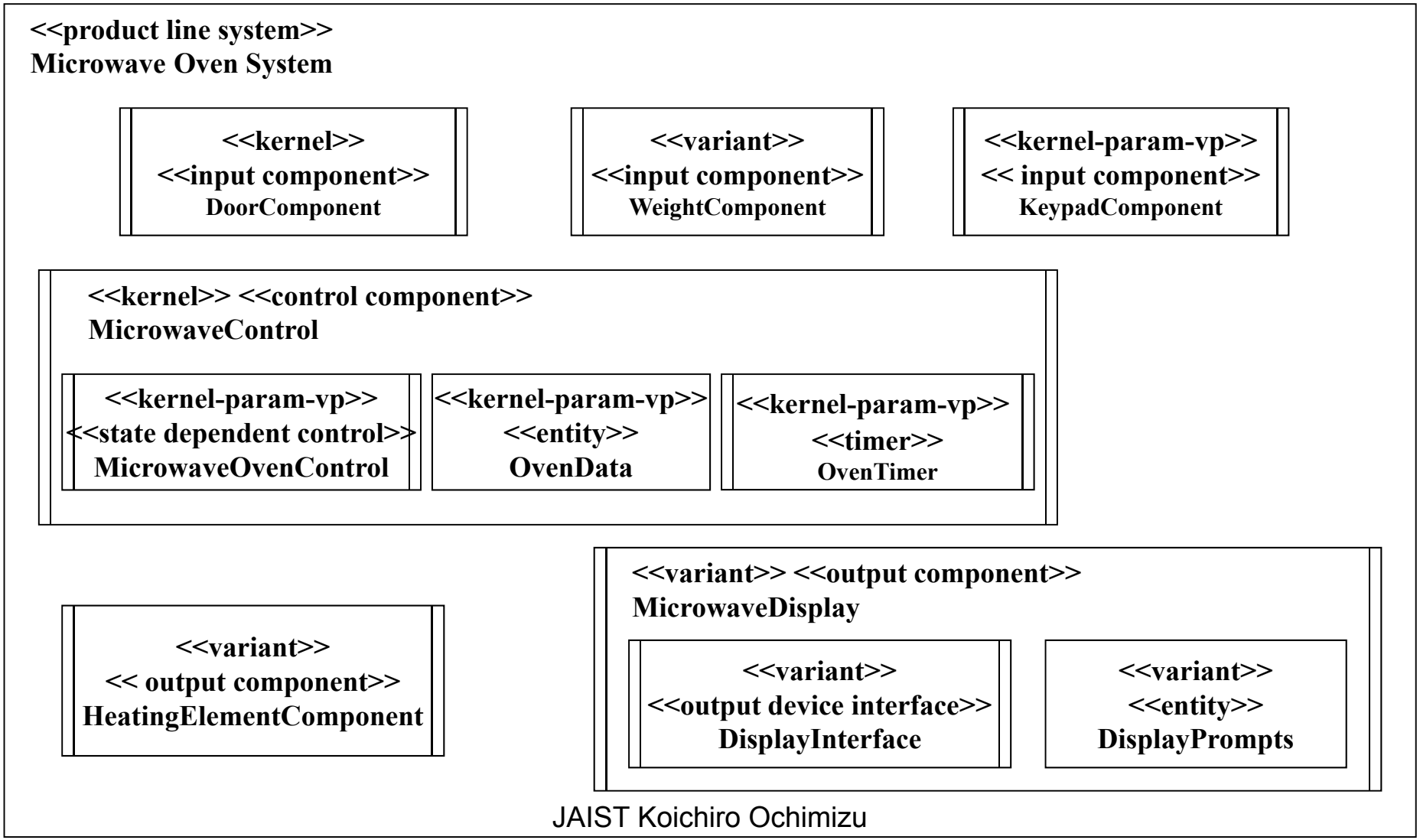


Design Modeling

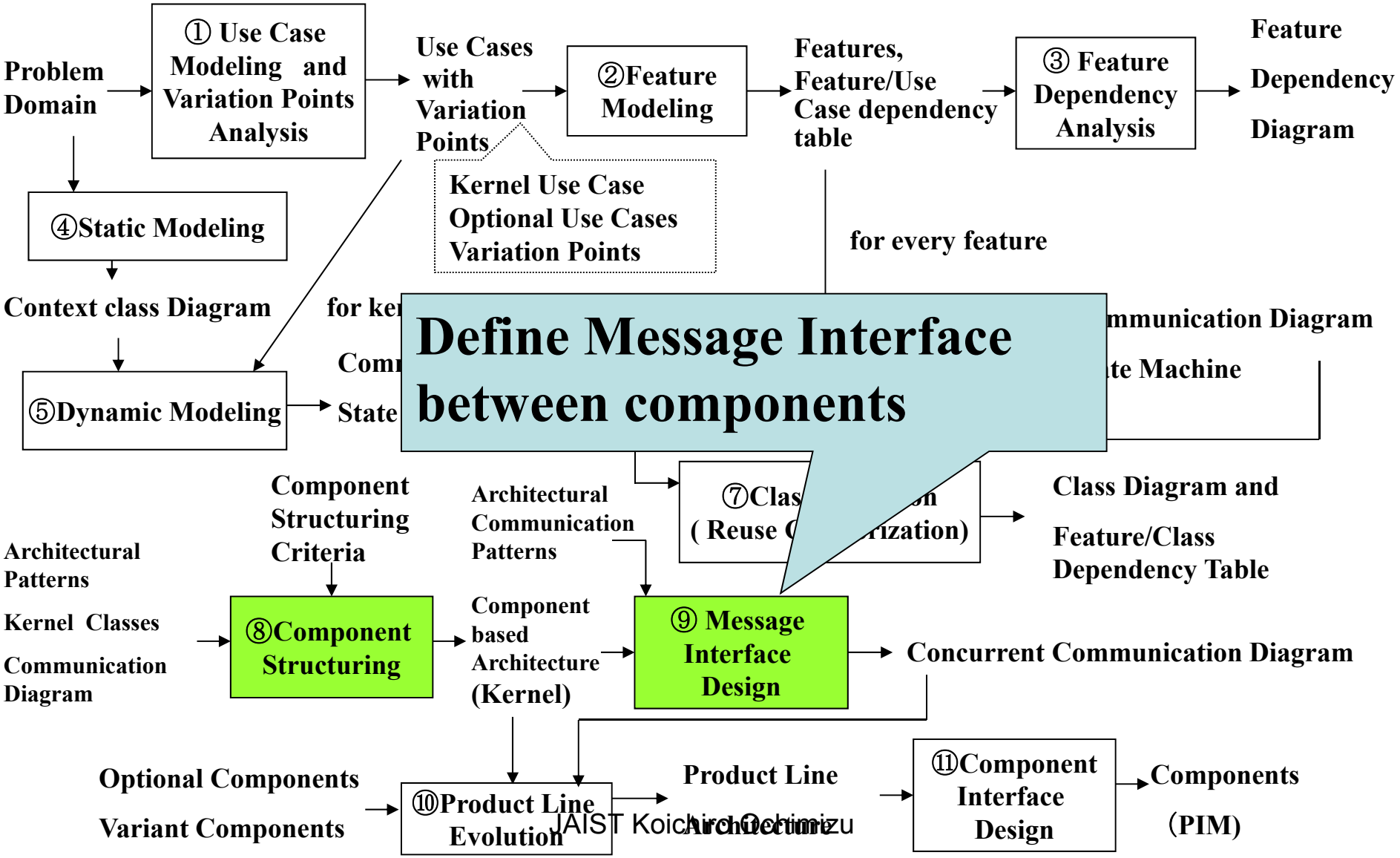
- The Microwave Oven software product line is designed as a component-based software architecture based on the Centralized Control pattern(One control component provides the overall control of the system, receiving messages from other components).
- The product line is designed as a distributed component-based software architecture.
- The component architecture is developed gradually
  - Starting with the design of the kernel system, which contains the kernel and default components
  - Next, the message communication between components is designed.
  - With the evolutionary design approach, this process is repeated for the full product line, at which point the optional and variant components are added.
- On the basis of the design of the overall product line component and communication architecture, the component ports and connectors are designed with the goal of maximizing component reuse in the different product line member configurations.
- Finally, the provide and required interfaces of each component are described.

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

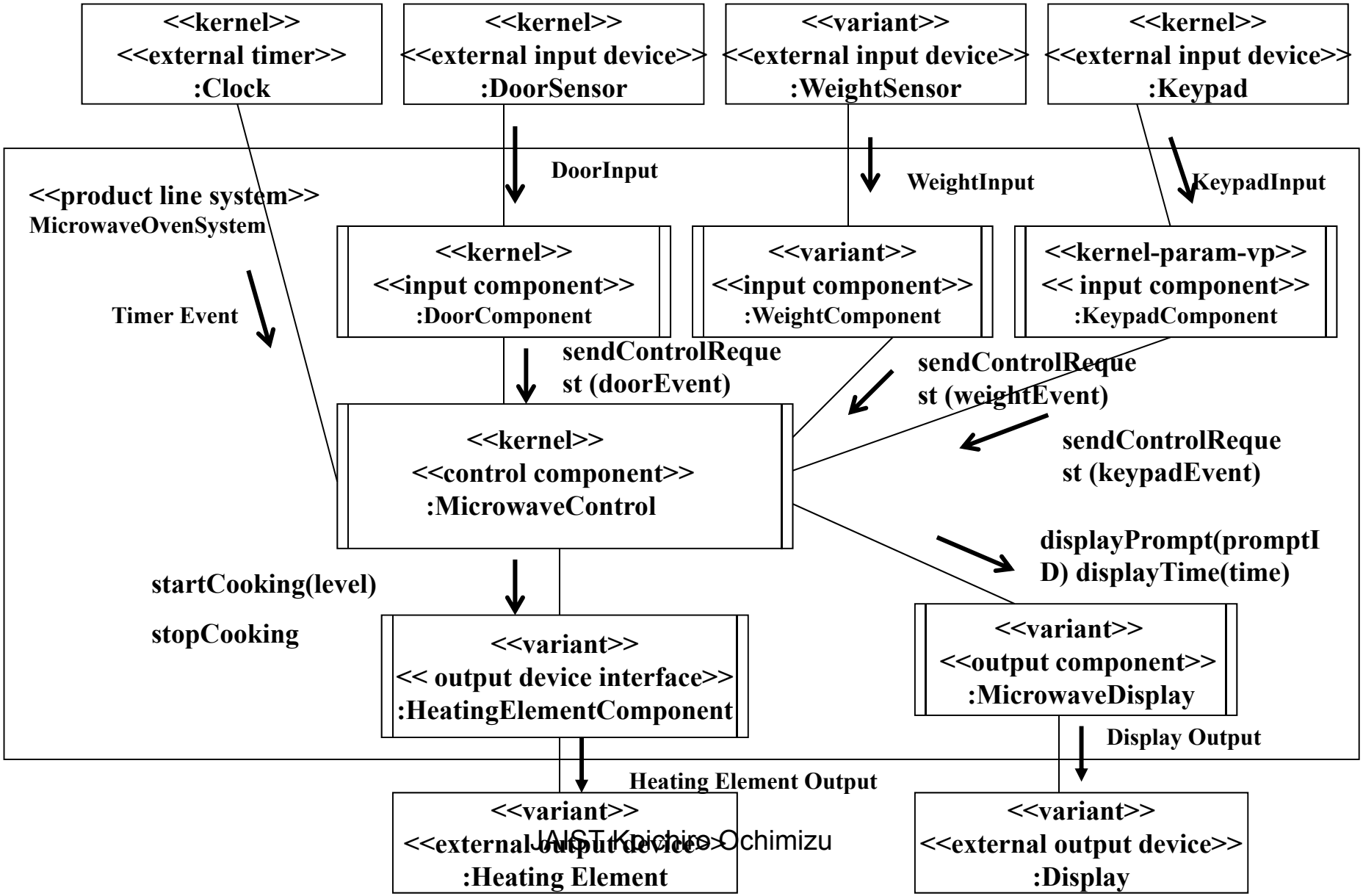
Component Structuring for Kernel



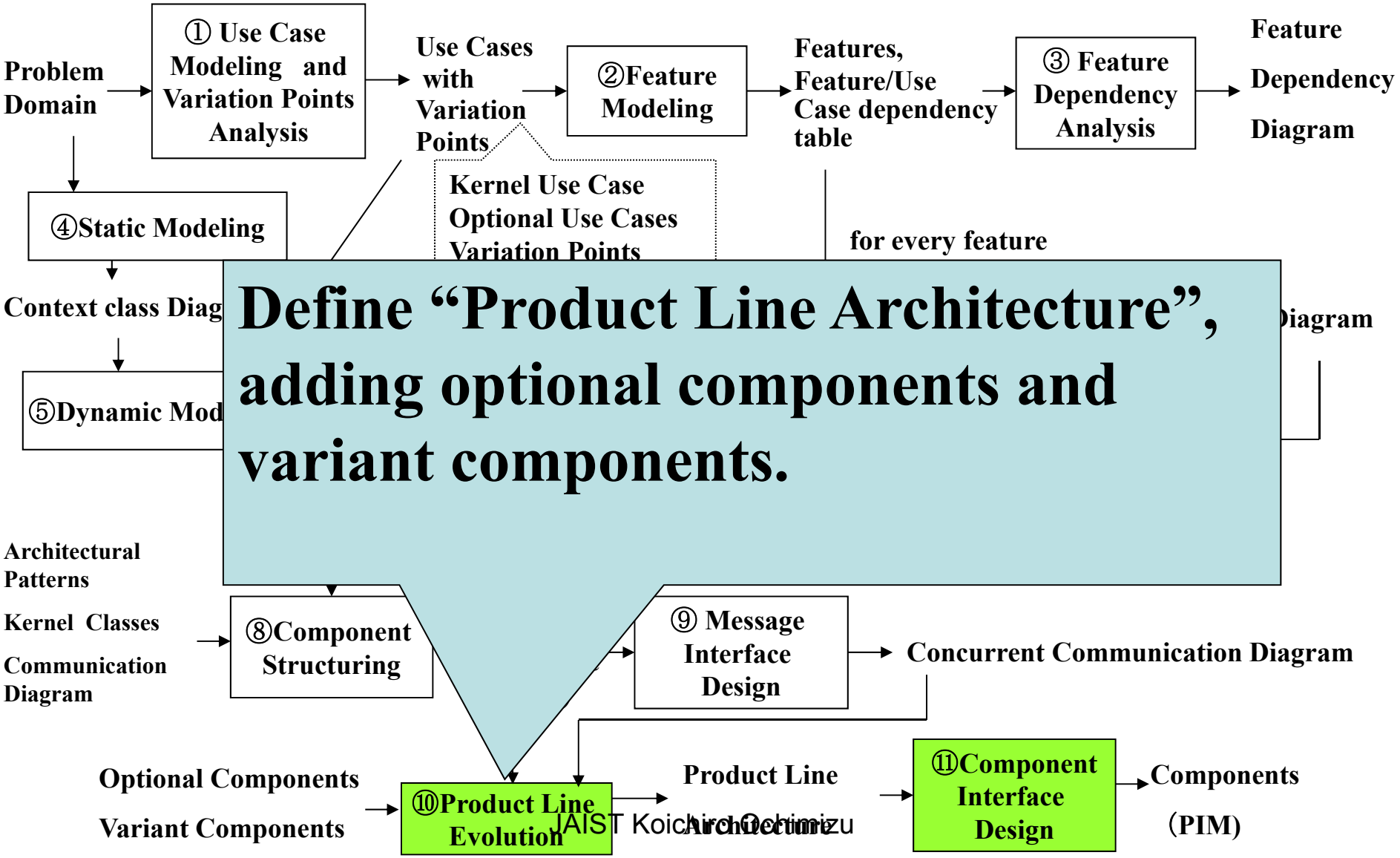
Outline of PLUS (Concurrent Communication Diagram)



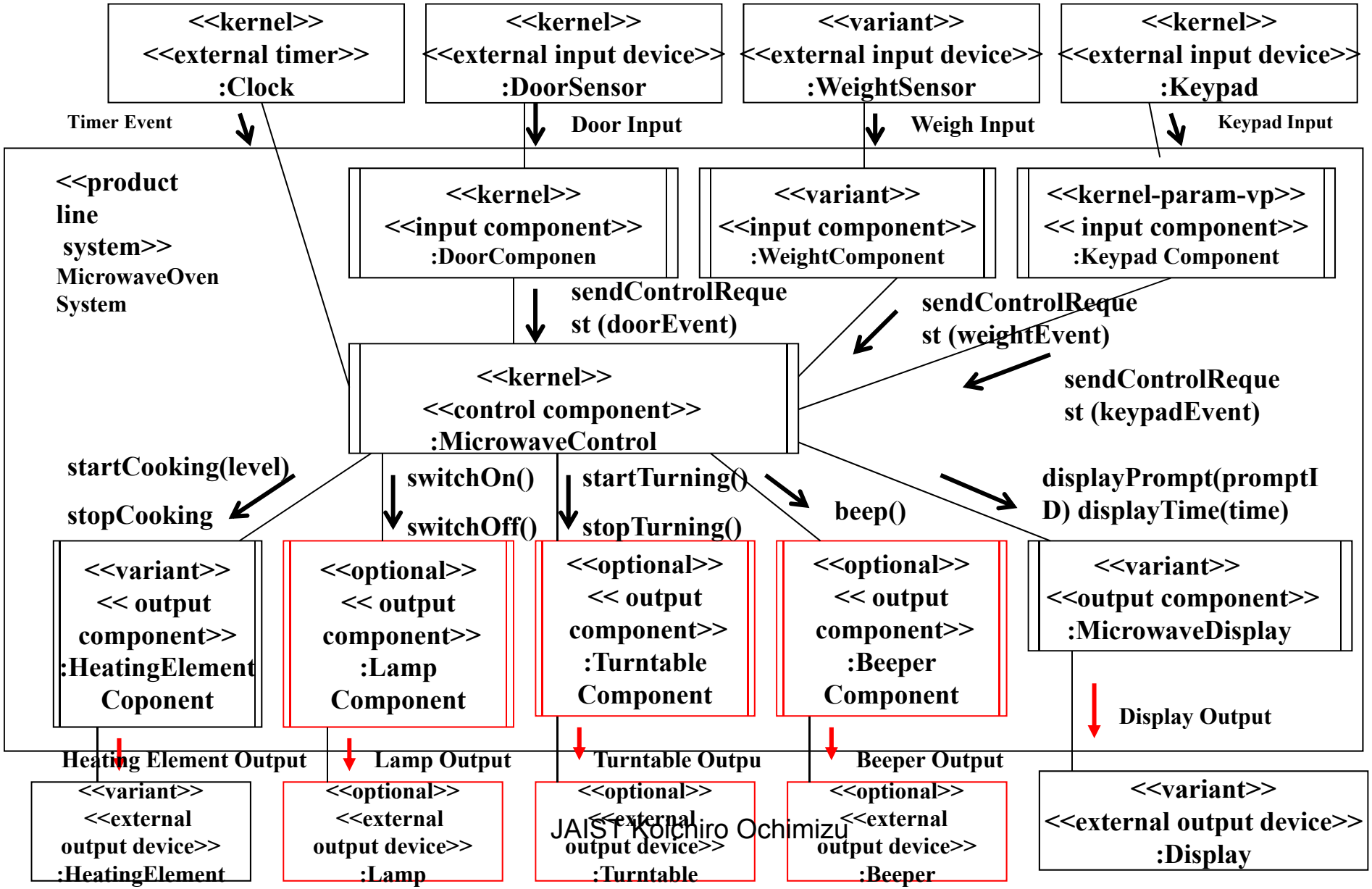
Distributed Software Architecture for Kernel: message Interface



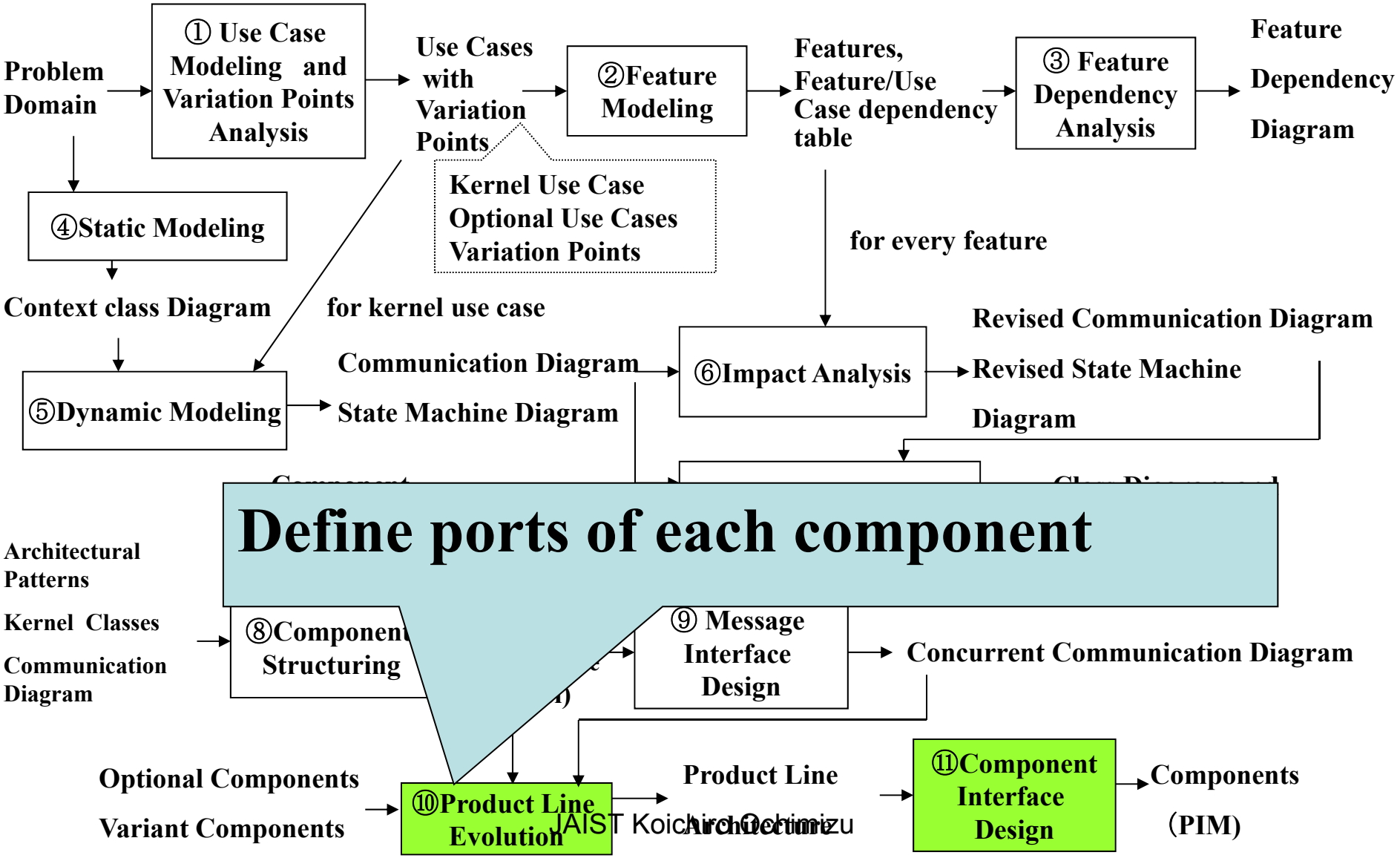
Outline of PLUS (Product Line Architecture)



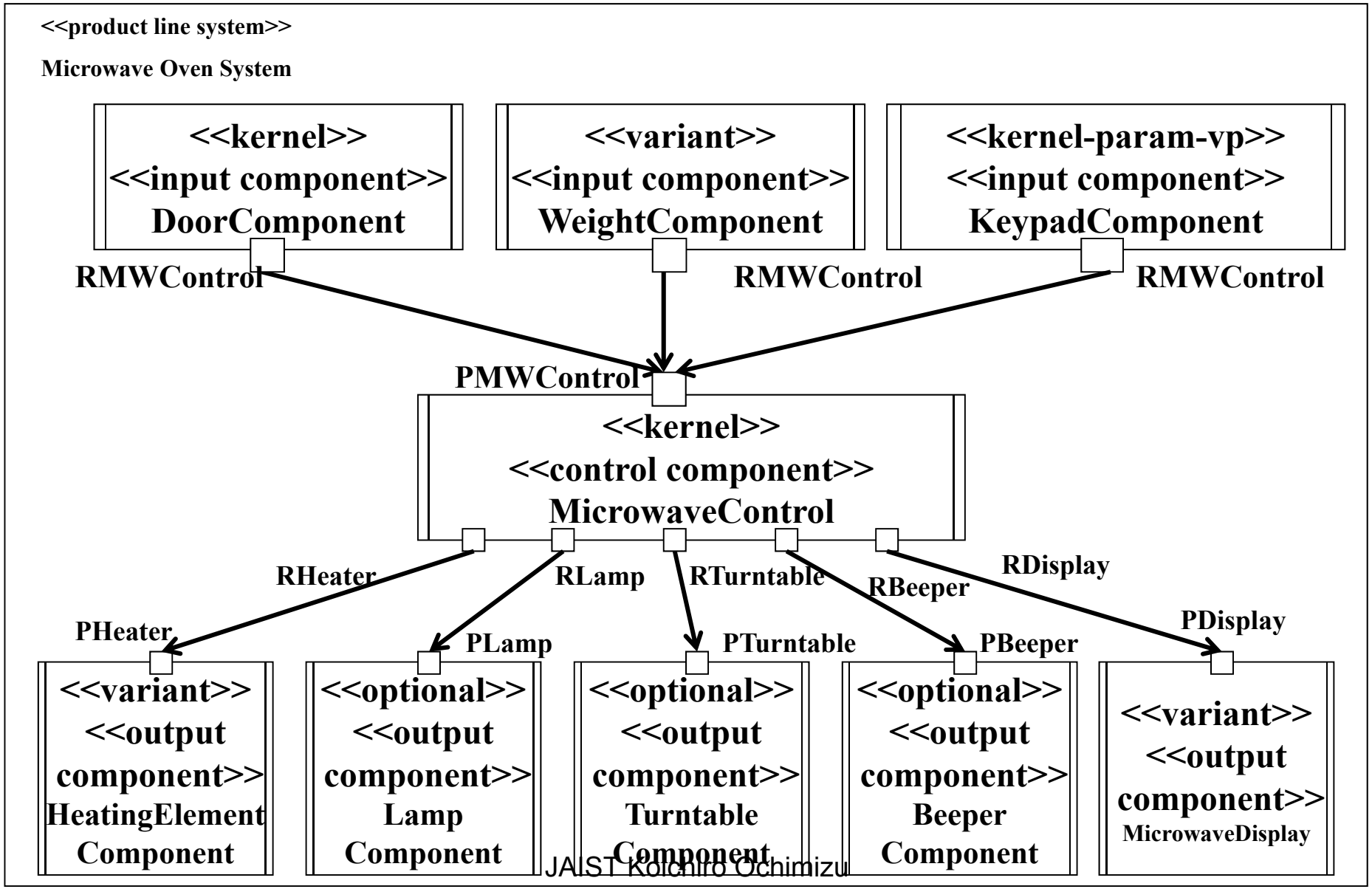
Distributed software architecture for the microwave oven software product line



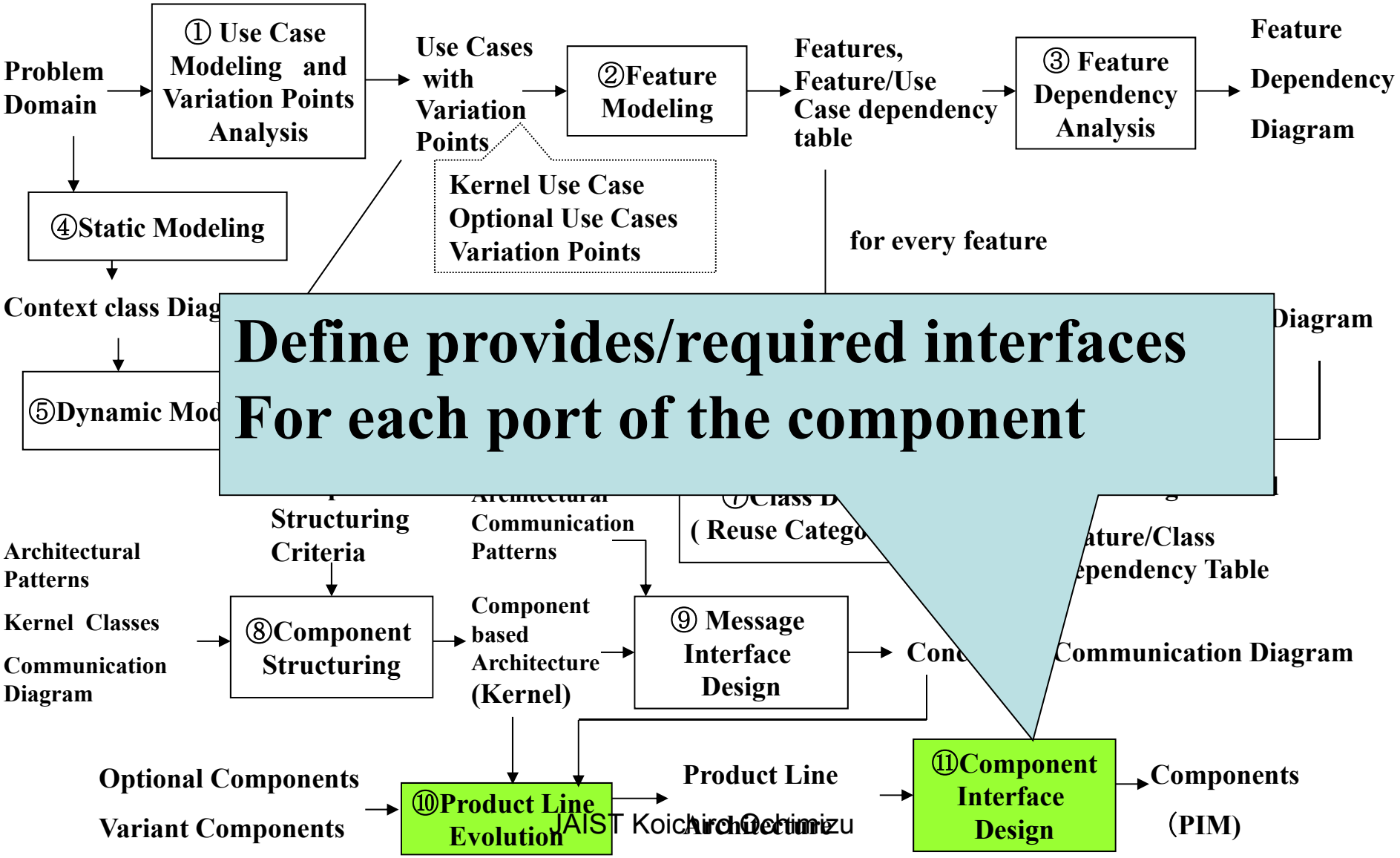
Outline of PLUS (Product Line Architecture)



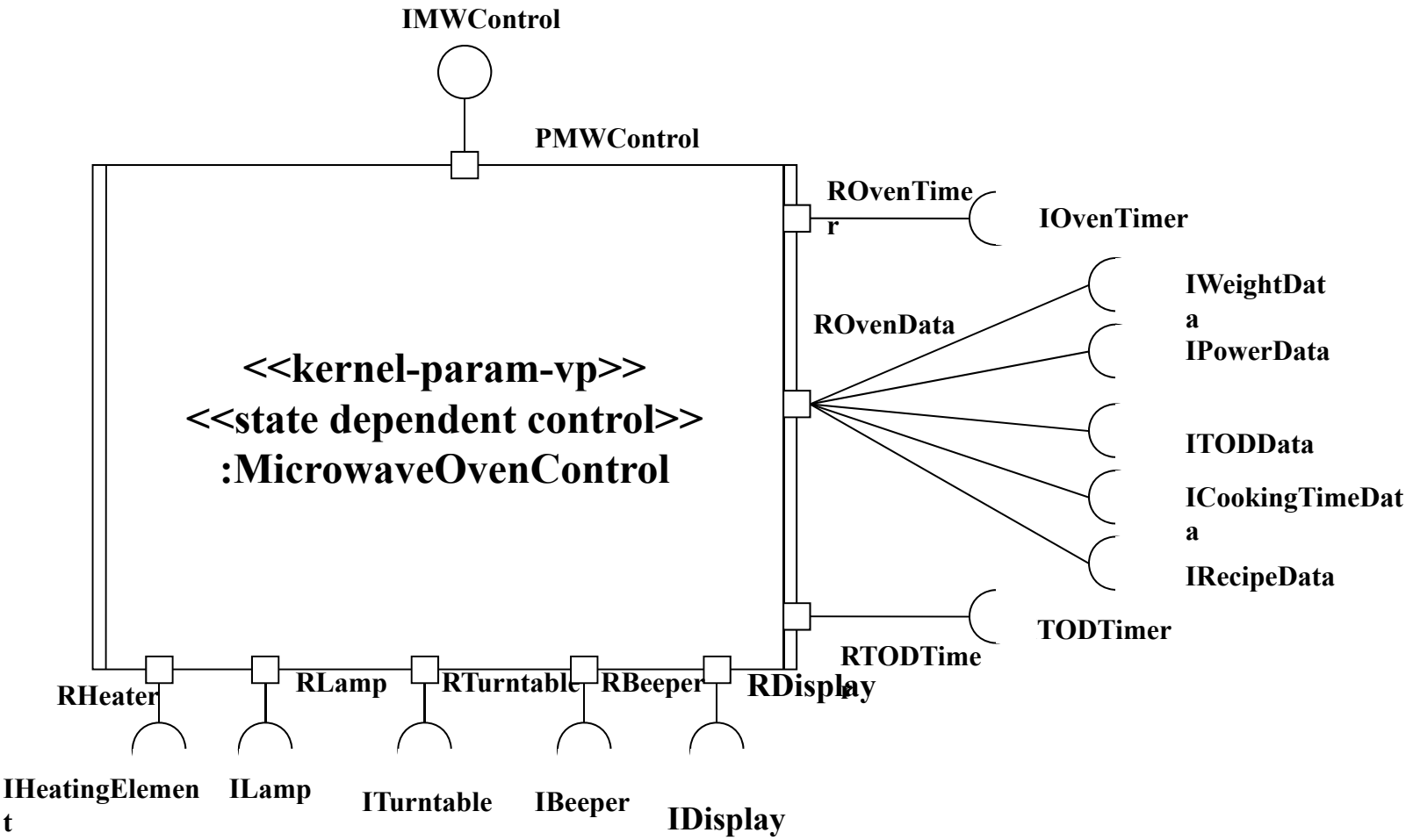
Microwave oven software product line architecture



Outline of PLUS (Component Interface Definition)

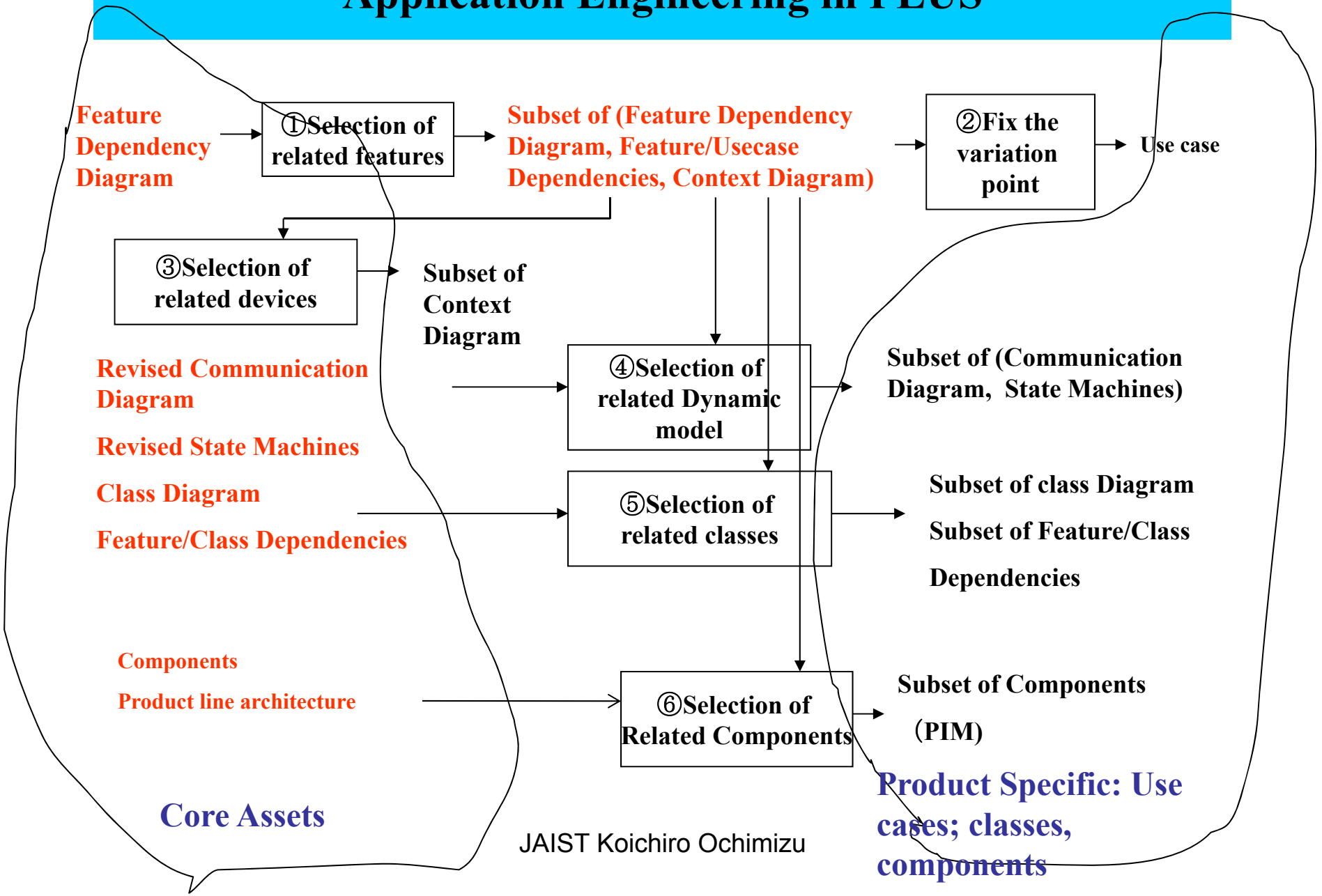


Ports and interfaces of the Microwave Oven Control component

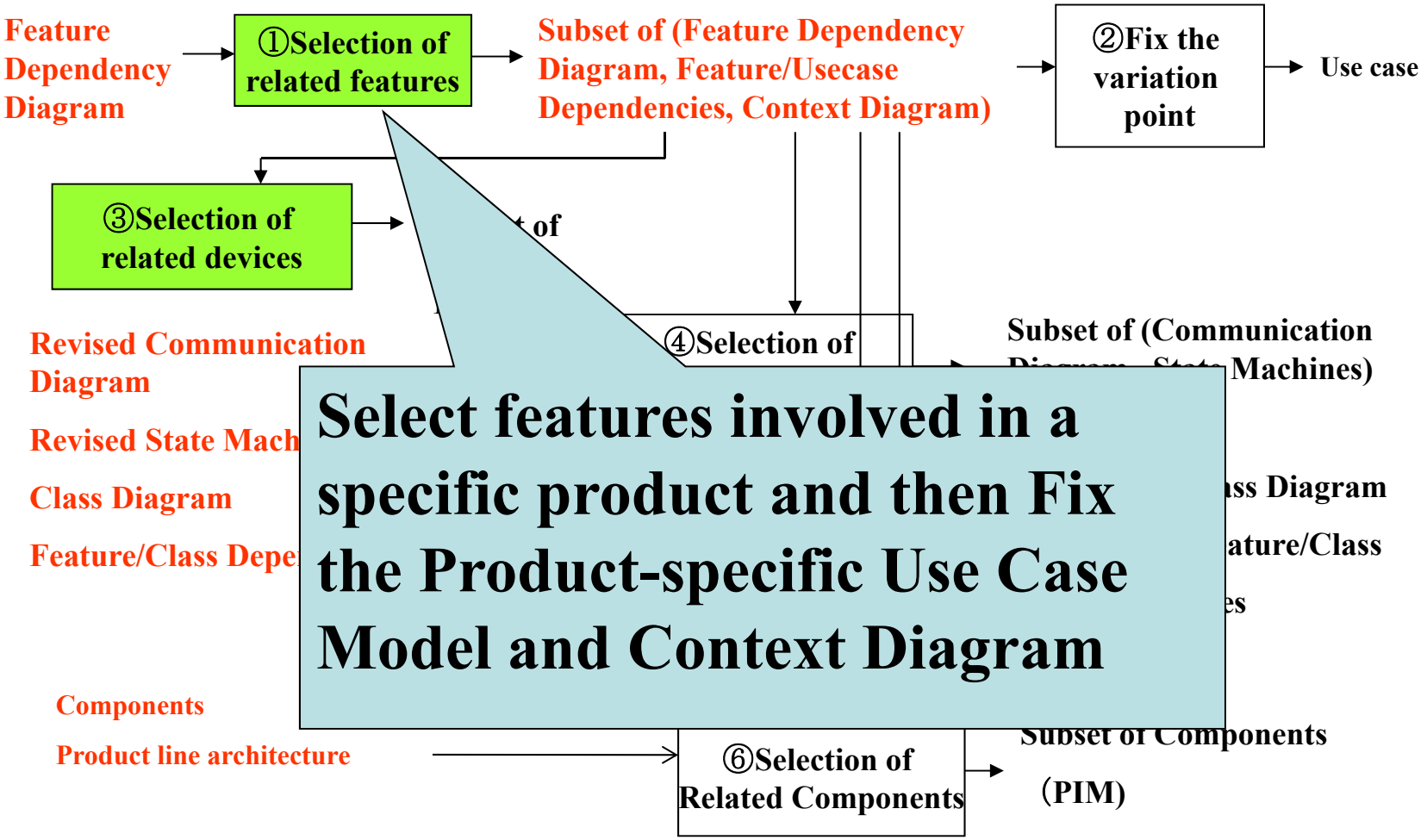


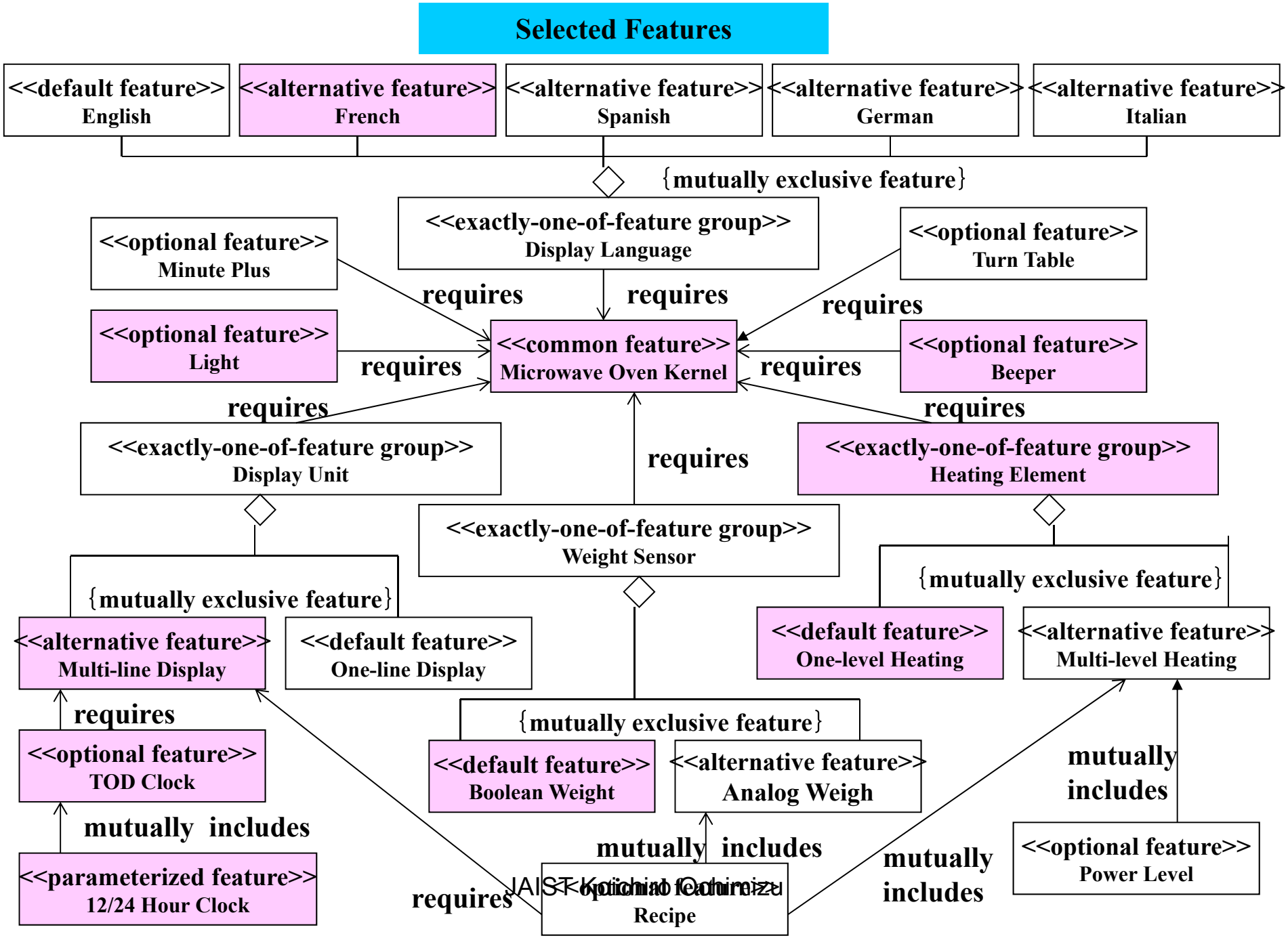
RBeeper RDisplay

Application Engineering in PLUS



Application Engineering in PLUS



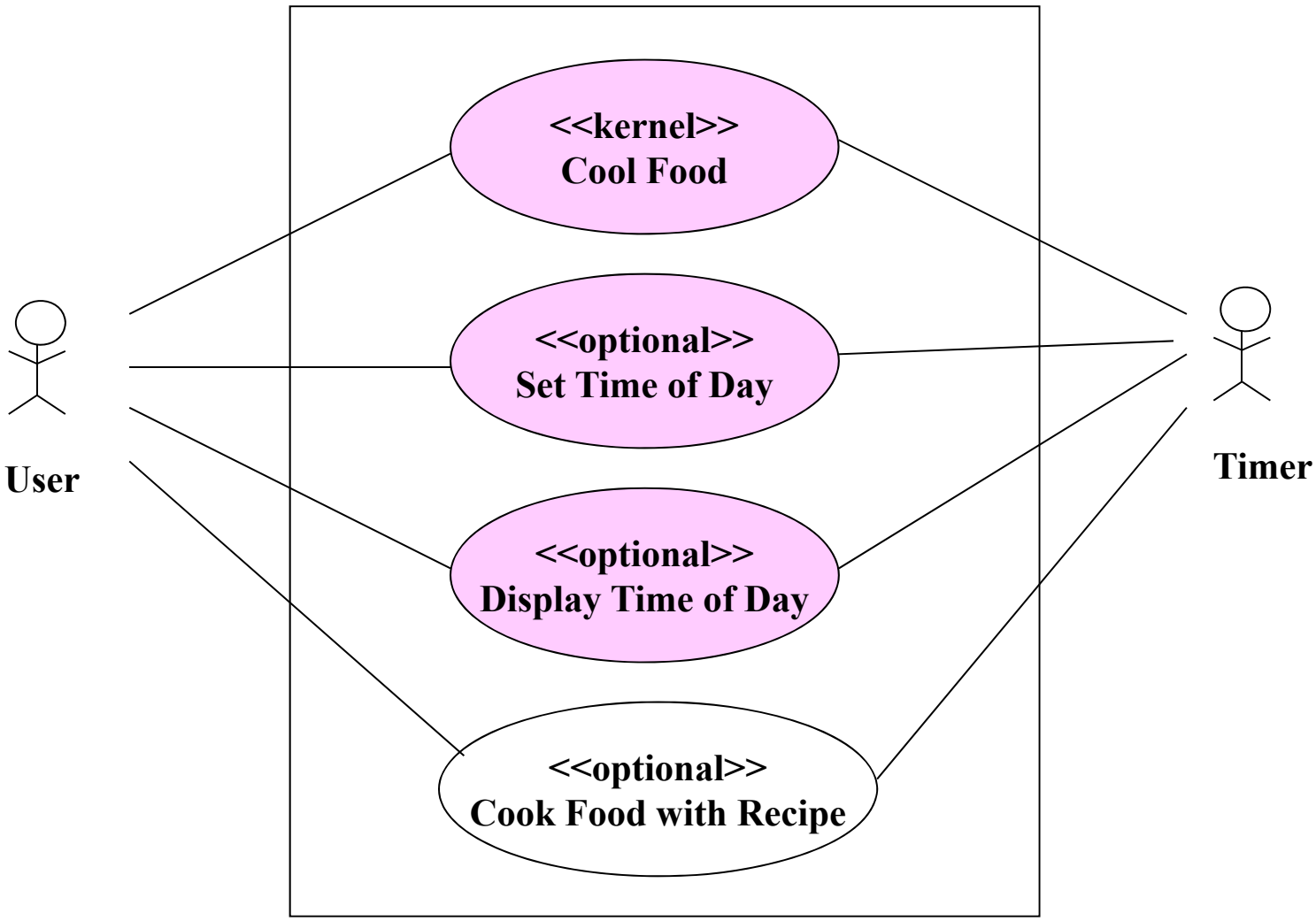

Selected Features
<<default feature>>
English
<<alternative feature>>
French
<<alternative feature>>
Spanish
<<alternative feature>>
German
<<alternative feature>>
Italian
{mutually exclusive feature}
<<exactly-one-of-feature group>>
Display Language
<<optional feature>>
Minute Plus
<<optional feature>>
Turn Table
<<optional feature>>
Light
<<common feature>>
Microwave Oven Kernel
<<optional feature>>
Beeper
requires
<<exactly-one-of-feature group>>
Display Unit
<<exactly-one-of-feature group>>
Heating Element
<<exactly-one-of-feature group>>
Weight Sensor
{mutually exclusive feature}
<<alternative feature>>
Multi-line Display
<<default feature>>
One-line Display
<<default feature>>
One-level Heating
<<alternative feature>>
Multi-level Heating
<<optional feature>>
TOD Clock
<<default feature>>
Boolean Weight
<<alternative feature>>
Analog Weigh
mutually includes
<<parameterized feature>>
12/24 Hour Clock
<<optional feature>>
Power Level
Recipe

Feature/Use Case Dependencies

Feature Name	Feature Category	Use Case Name	Use Case Category/ Variation Point (vp)	Variation Point Name
Microwave Oven Kernel	common	Cook Food	Kernel	
Light	optional	Cook Food	VP	Light
Turn Table	optional	Cook Food	VP	Turn Table
Beeper	optional	Cook Food	VP	Beeper
Minute Plus	optional	Cook Food	VP	Minute Plus
One-line Display	default	Cook Food	VP	Display Unit
Multi-line Display	alternative	Cook Food	VP	Display Unit

Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

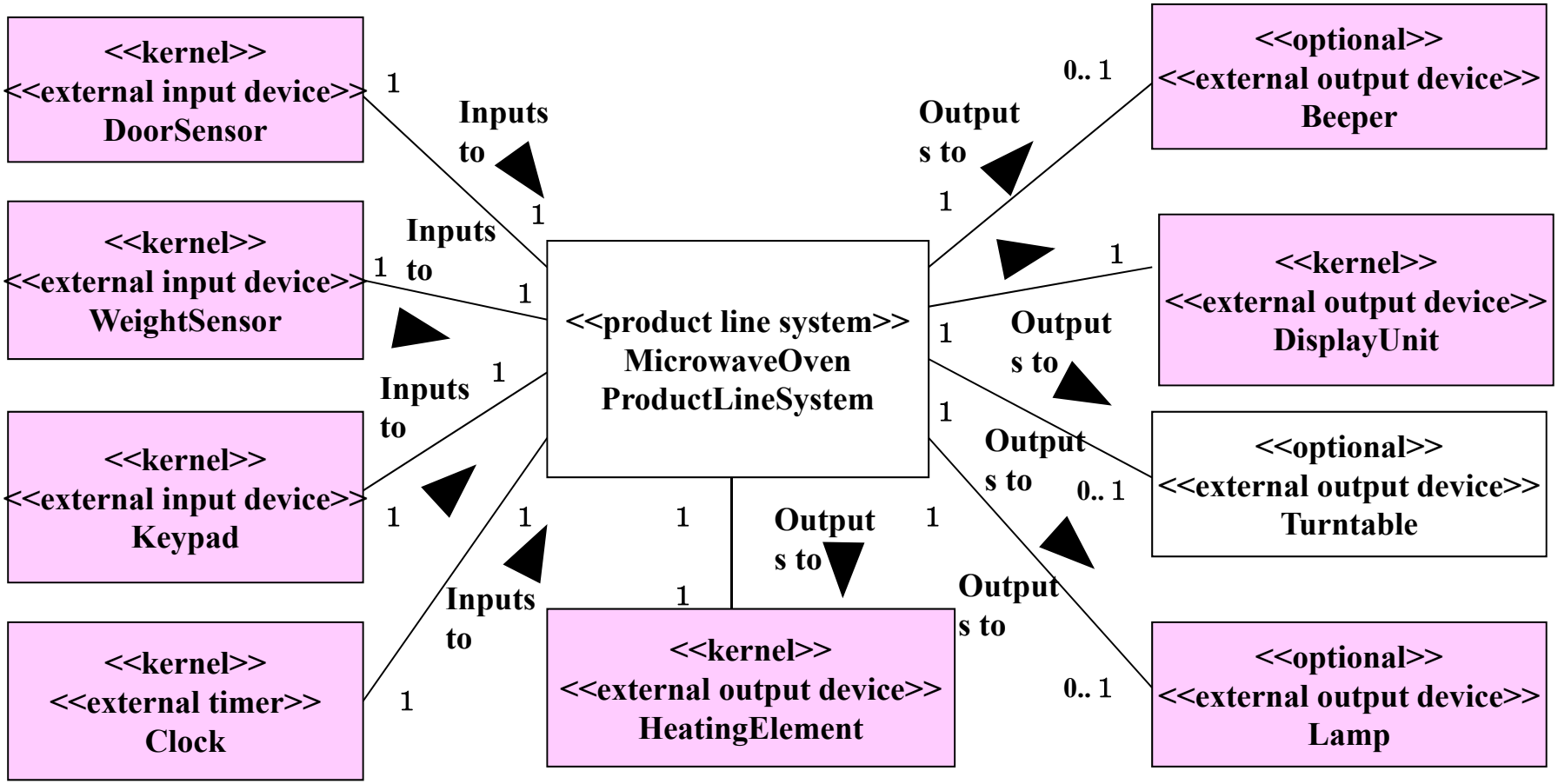
Selected Use Cases



Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

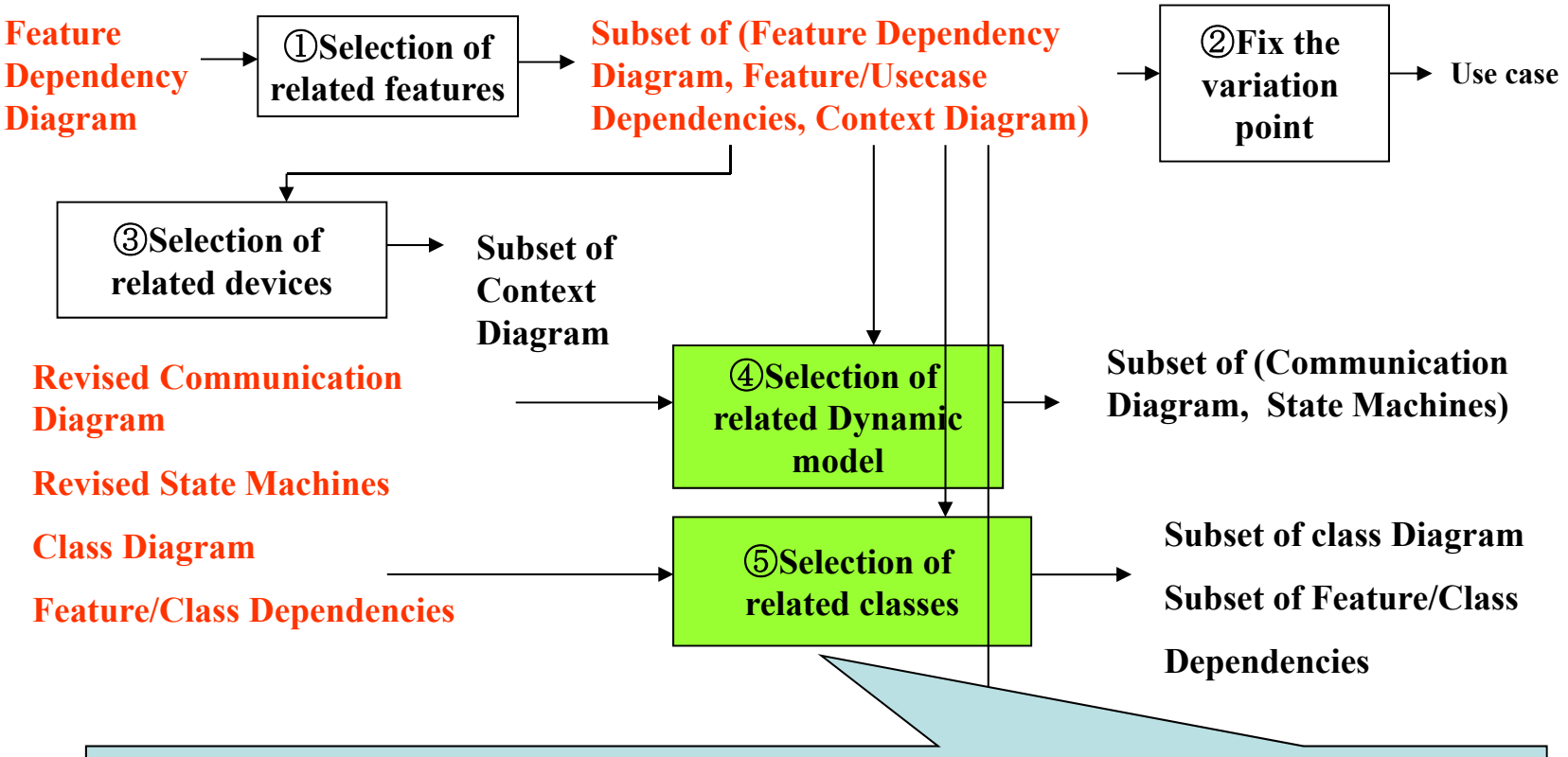
Selected External Environments

- The product line context class diagram defines the boundary between a product line system (i.e. any member of the product line) and the external environment(i.e. the external classes to which members of the product line have to interface)



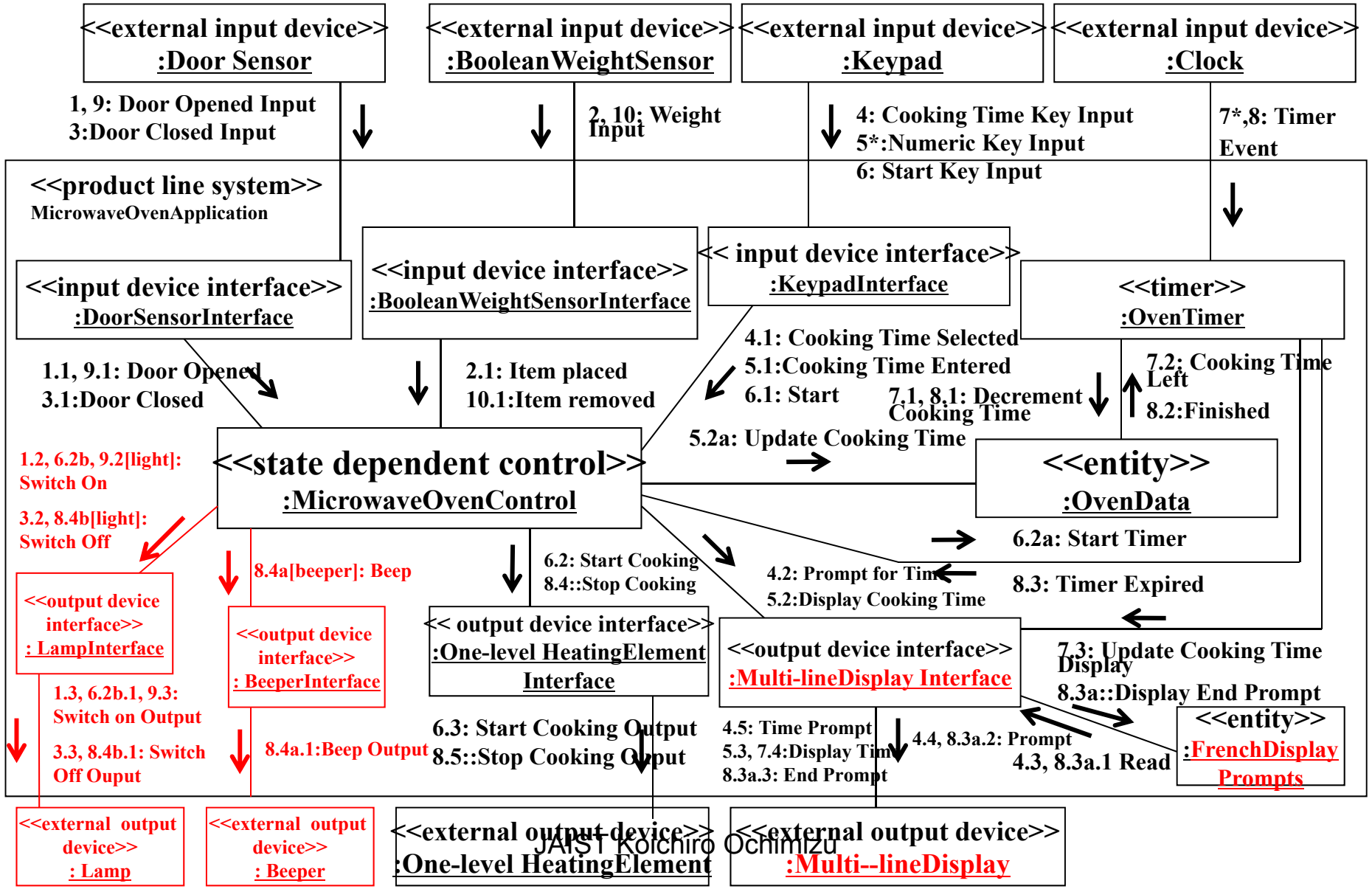
Hassan Gomaa, “Designing Software Product Lines with UML”, Addison-Wesley, 2005.

Application Engineering in PLUS



Customize the dynamic models(communication diagram and state machines) and the related classes

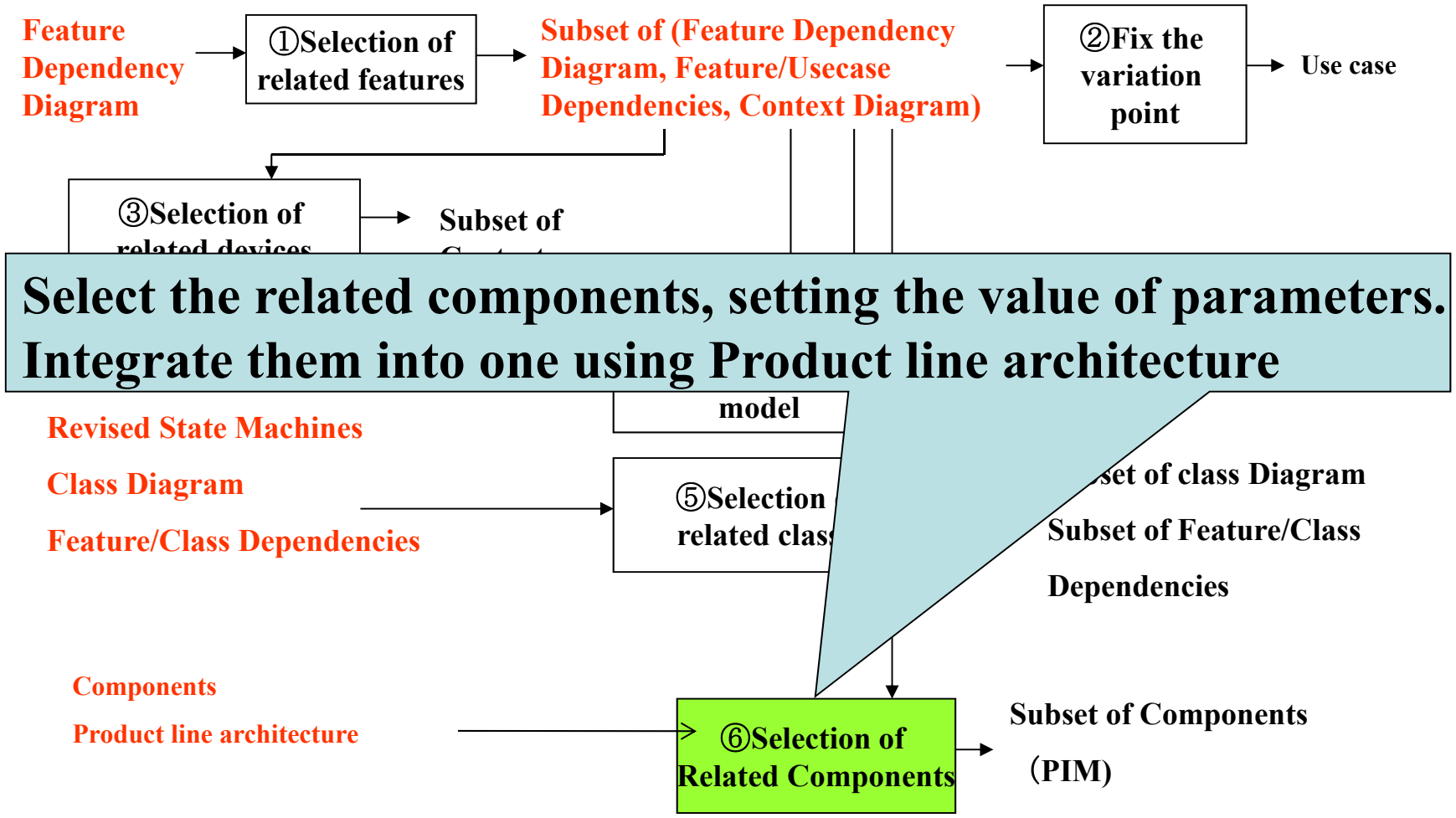
Communication Diagram for the application



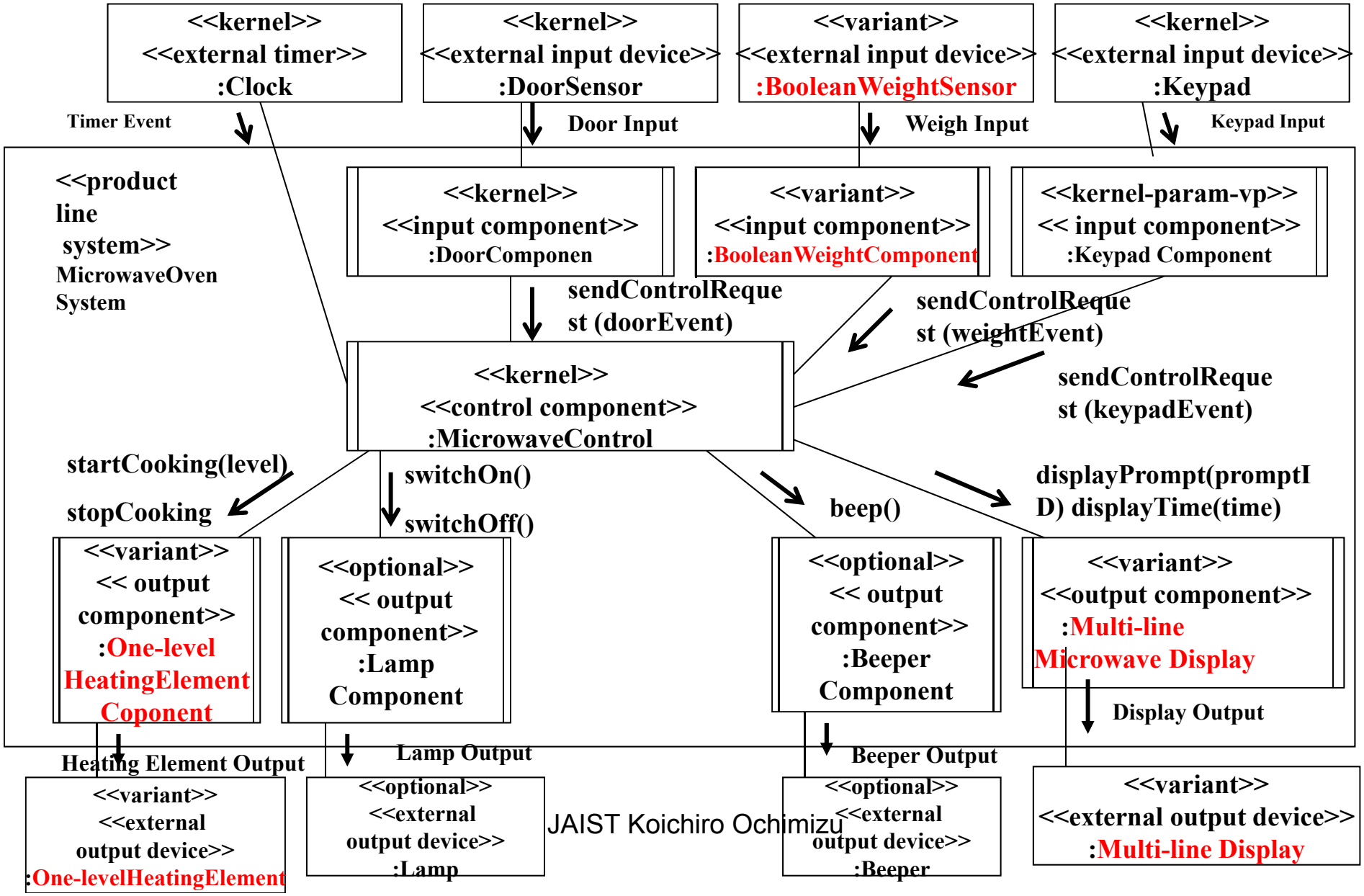
Feature/class Dependencies

Feature Name	Feature Category	Class Name	Class Category	Class Parameter
Light	optional	Lamp Interface	optional	
		Microwave Oven Control	kernel-param-vp	light:Boolean
Turntable	optional	Turntable Interface	optional	
			kernel-param-vp	turntable:Boolean
Beeper	optional	Beeper Interface	optional	
		Microwave Oven Control	kernel-param-vp	beeper:Boolean
Minute Plus	optional	Keypad Interface	kernel-param-vp	minuteplus:Boolean
		Microwave Oven Control	kernel-param-vp	minuteplus:Boolean
		Oven Timer	kernel-param-vp	minuteplus:Boolean
		Oven Data	kernel-param-vp	minuteplus:Boolean

Application Engineering in PLUS



Distributed software architecture for the microwave oven software product line



Evaluation of PLUS

- PLUS provides with the well-organized way to determine features, classes, and components and to define the clear relationships among them.
- PLUS only supports to make a core asset from scratch. It does not support to build the core asset by mining legacy software or purchasing COTS