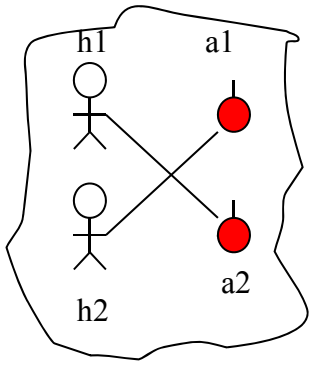
Content(2)

- **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

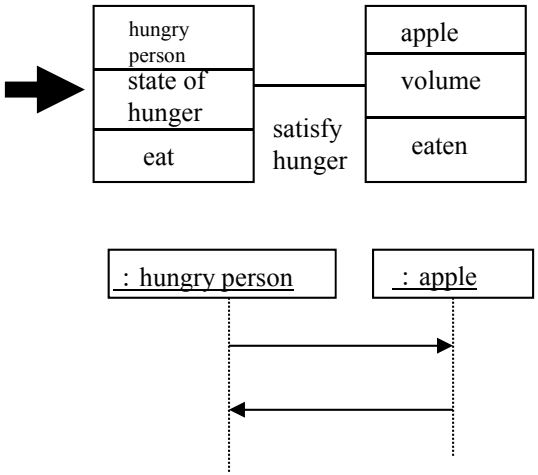
Four worlds in OOT

Abstraction of entities in the domain from the viewpoint of satisfy-one's-hunger



The world view : We can represent the domain simply by "A set of objects and their interaction)

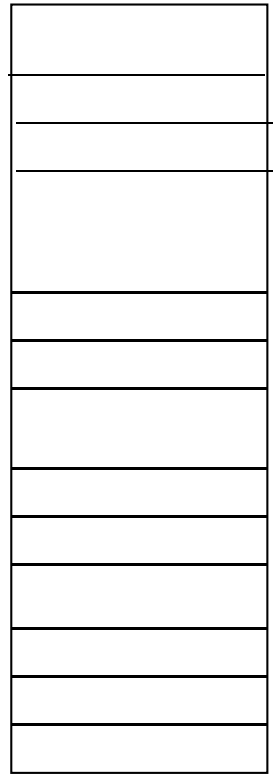
Description of possibility



Definition of static structure and dynamic behavior

```
public class hungry-person {
 int state-of-hunger
 void eat ()
}
public class apple {
 int volume
 void eaten ()
}
```

Program

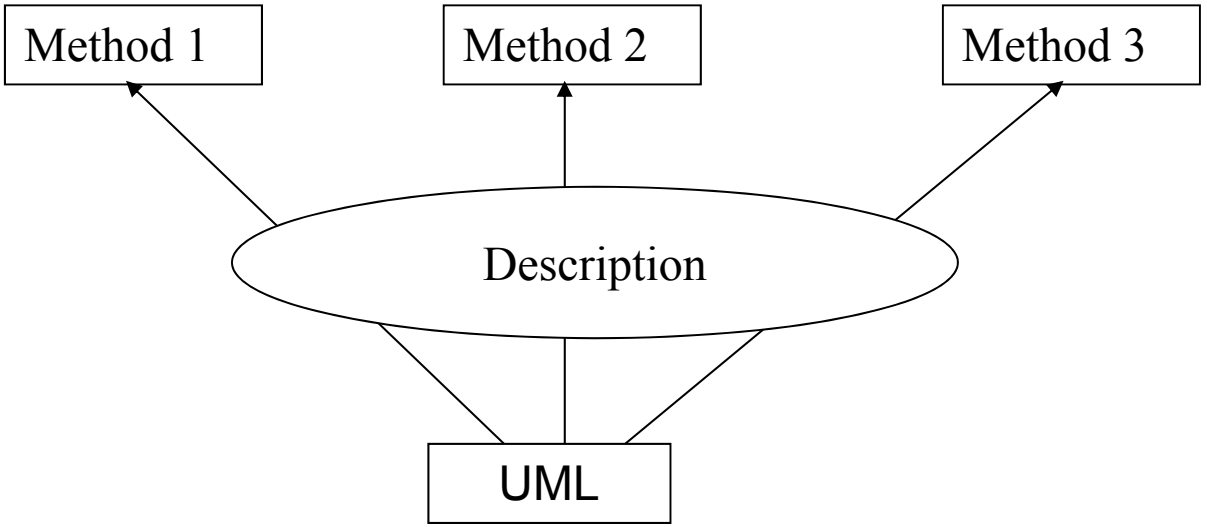


Reproduction of the Domain in the main memory

abstraction | specification

implementation | instanciation

Relationship between Methods and UML



Diagrams of UML are used for

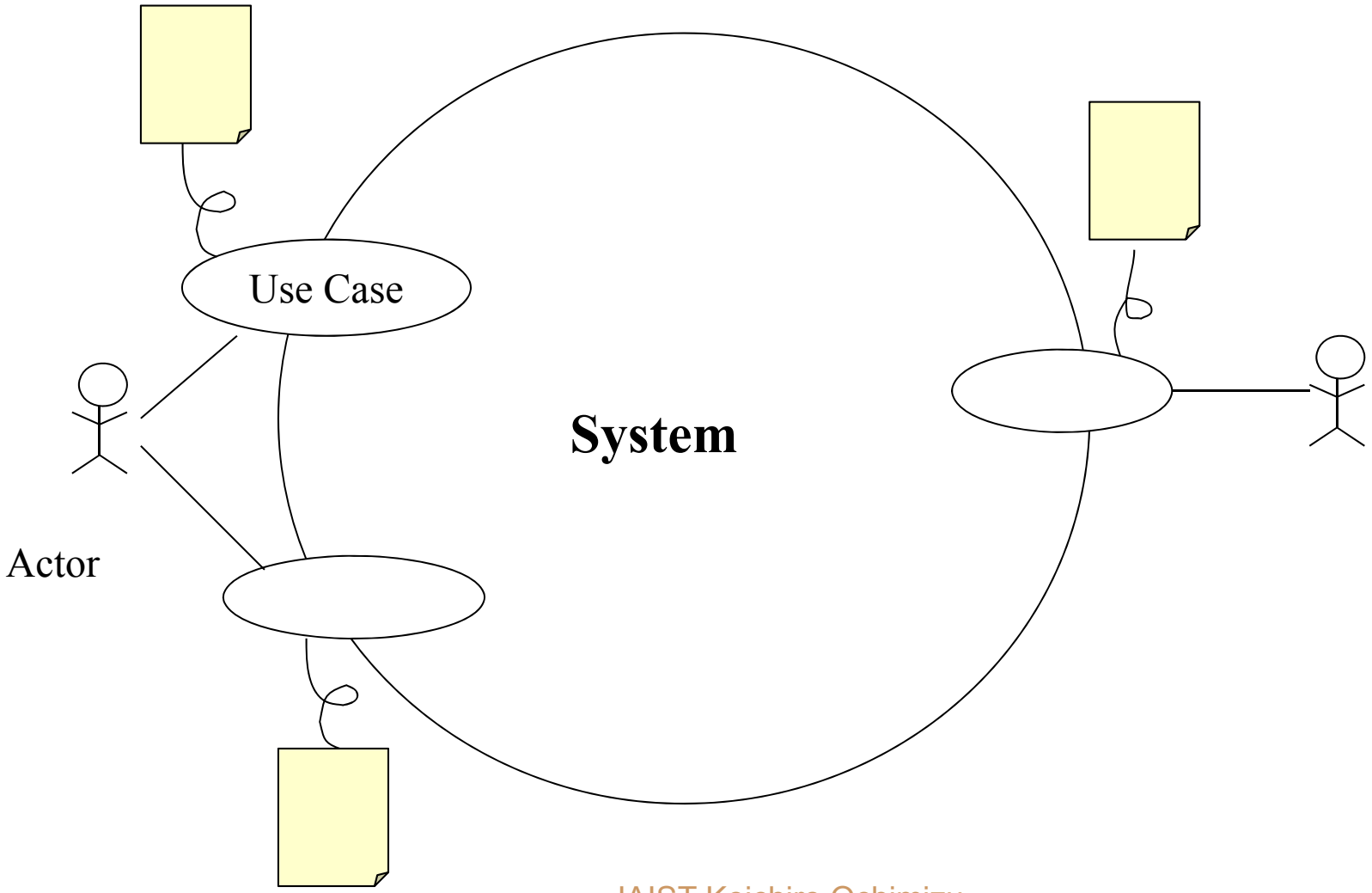
- Very popular now and help us make and analyze:
  - Use-case Diagrams for defining functional requirements
  - Communication Diagrams for finding analysis classes
  - Class Diagrams for designing the static structure
  - Sequence Diagrams for defining objects interaction
  - State Diagrams for defining the behavior of each object
  - Deployment Diagrams for allocating objects to machines
  - Component Diagrams for packaging

Use-case Driven approach

Use Case Description

Event Sequences between actors and the system

Functional Requirements



Use Case Model

Use Case Model : A use case model represents the functional requirements and consists of actors and use cases. A use case model helps the customer, users, and developers agree on how to use the system.

Actor: An actor is someone or something that interacts with system.

System: Black box provided with use cases

Use Case: A use case specifies a sequence of actions that the system can perform and that yields an observable result of value to a particular actor.

I. Jacobson, G.Booch, J.Rumbaugh,"The Unified Software Development Process", Addison Wesley, 1999.

What is an Actor ?

- An actor is someone or something that interacts with the system.
- The actor is a type (a class), not an instance.
- The actor represents a role, not an individual user of the system.
- Actors can be ranked. A primary actor is one that uses the primary functions of the system. A secondary actor is one that uses secondary functions of the system, those functions that maintain the system, such as managing data bases, communication, backups, and other administration tasks.

H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

What is a Use Case ?

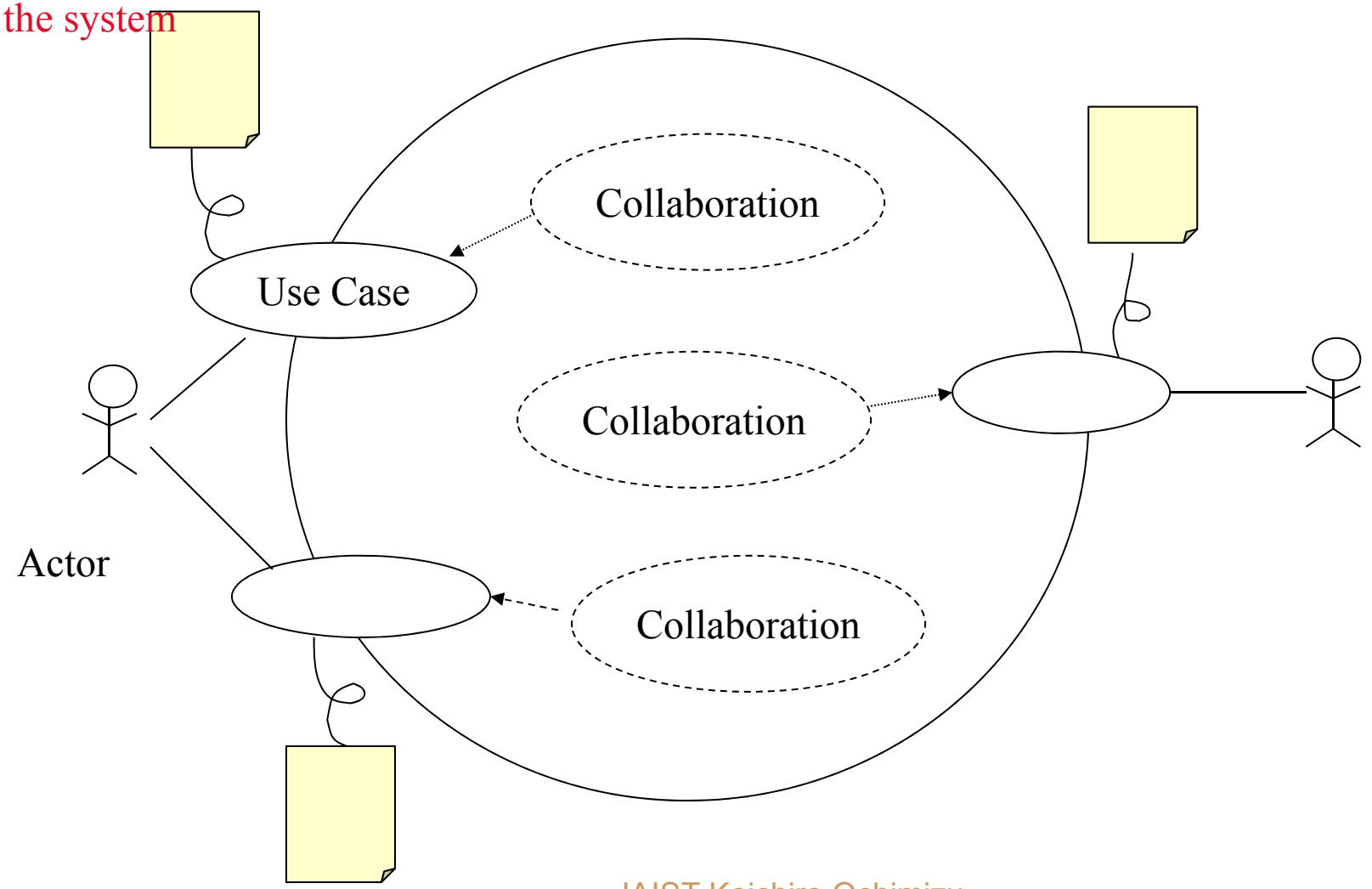
- A use case represents a complete functionality as perceived by an actor.
- A use case is always initiated by an actor.
- A use case provides values to an actor.
- Use cases are connected to actors through associations (communication association).

H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

Analysis of inside of the system

Use Case Description

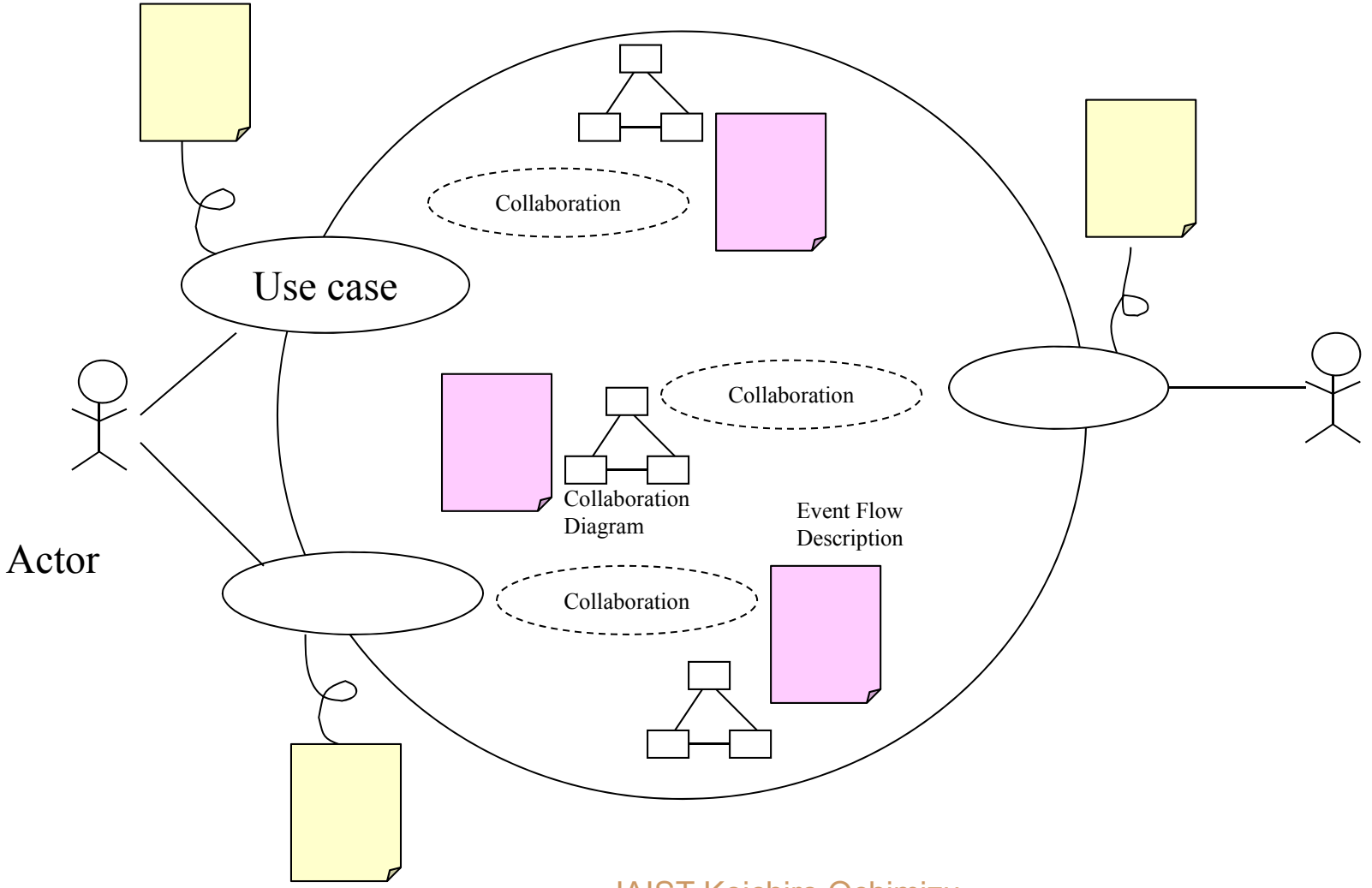
Event Sequences between actors and the system



Analysis Classes

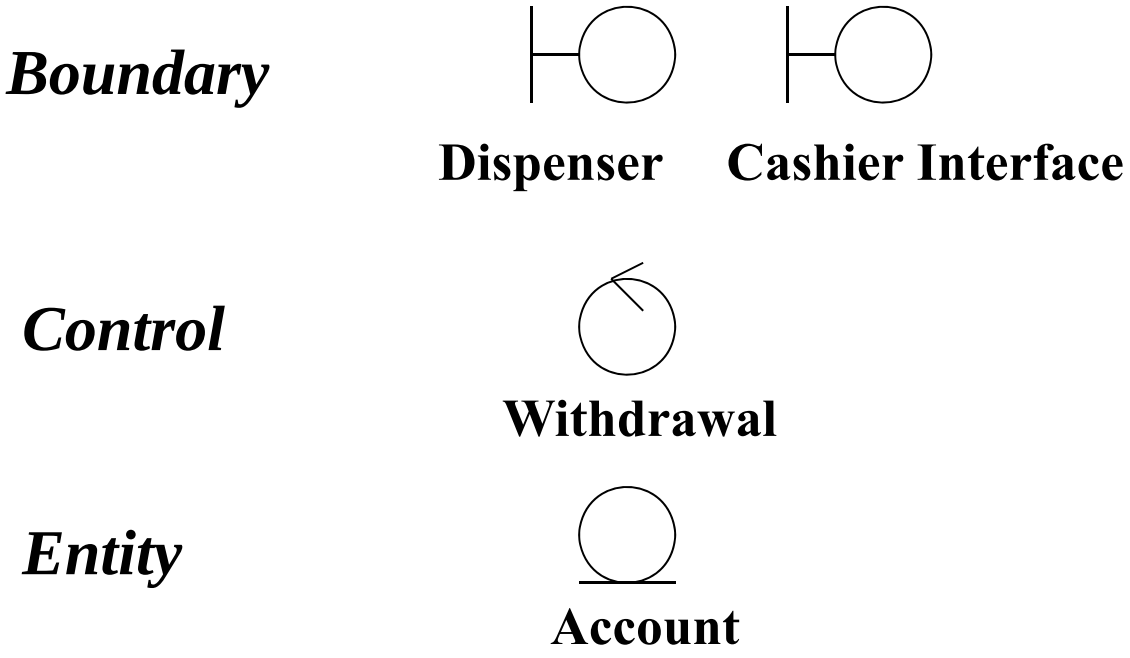
Use Case Description

Event Sequences between actors and the System



Analysis Stereotypes

In the analysis model, three different stereotypes on classes are used: <<boundary>>, <<control>>, <<entity>>.



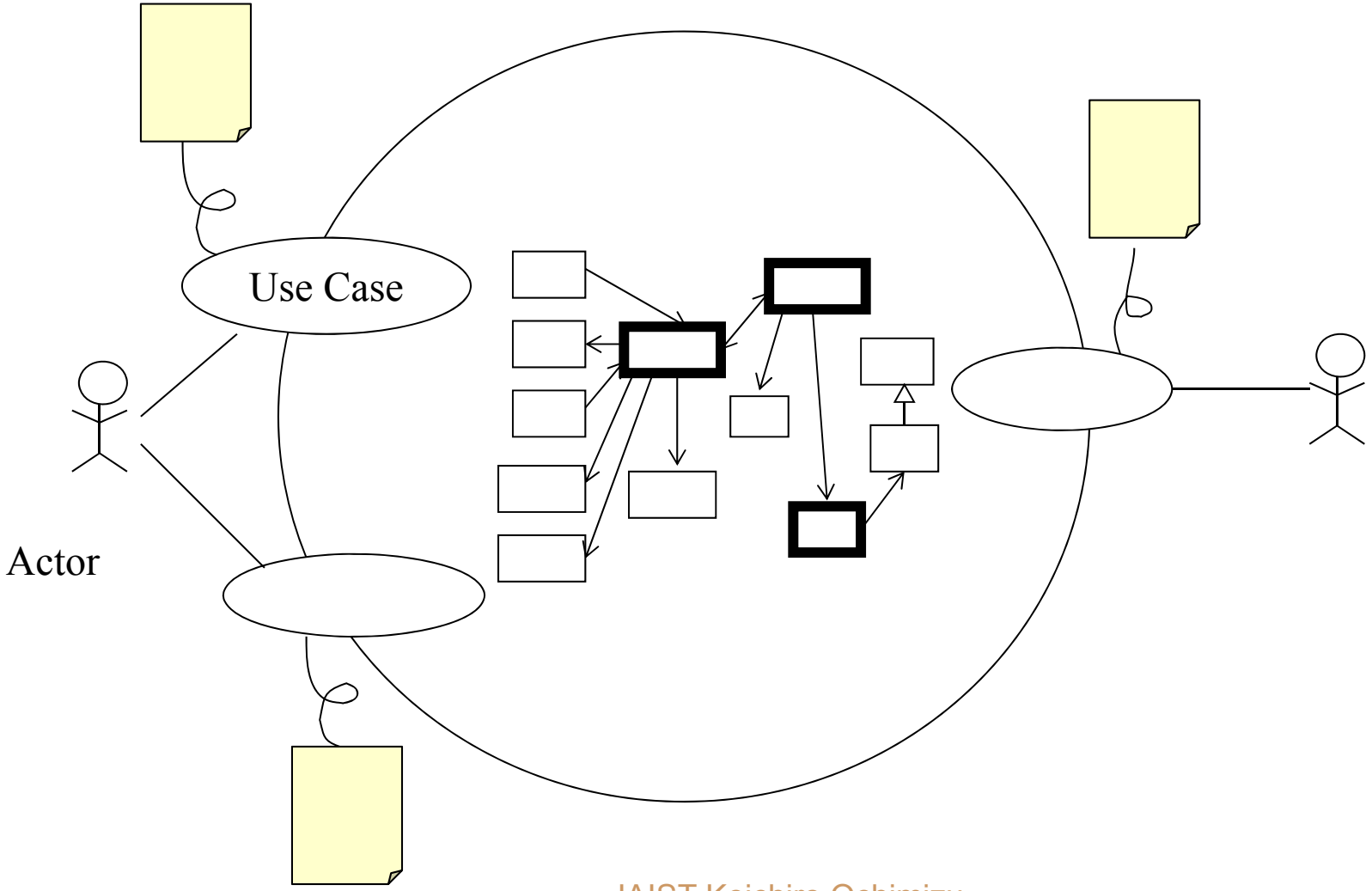
I. Jacobson, G.Booch, J.Rumbaugh,"The Unified Software Development Process", Addison Wesley, 1999.

Analysis Stereotypes

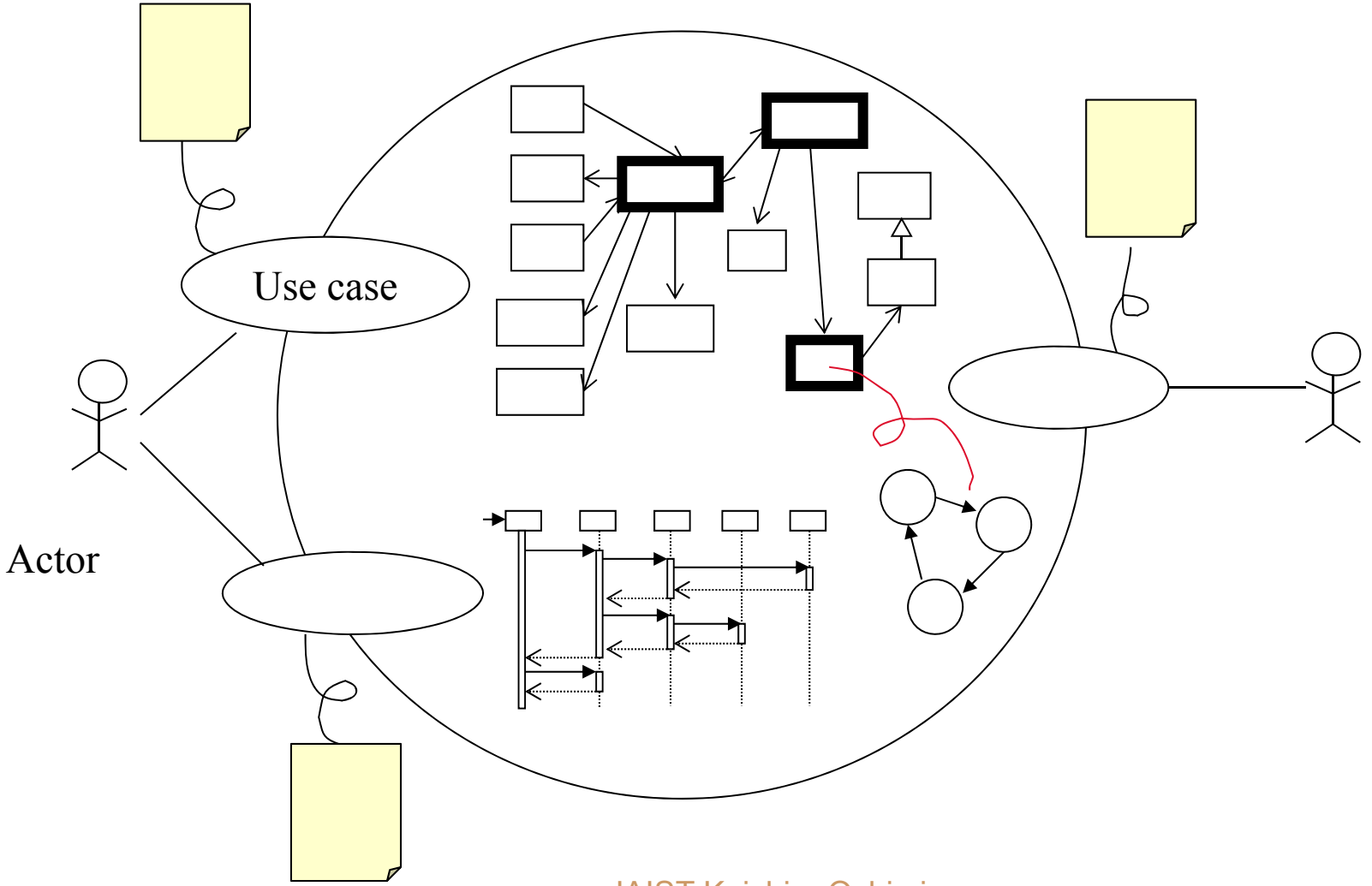
- <<boundary>> classes in general are used to model interaction between the system and its actors.
- <<entity>> classes in general are used to model information that is long-lived and often persistent.
- <<control>> classes are generally used to represent coordination, sequencing, transactions, and control of other objects. And it is often used to encapsulate control related to a specific use case.

I. Jacobson, G.Booch, J.Rumbaugh,"The Unified Software Development Process", Addison Wesley, 1999.

Class Diagram (Analysis Class + Design Class)



Final Step of Modeling (Definition of Static Structure and Dynamic Behavior)



Exercise

- **Review the content of my lecture by answering the following simple questions. Please describe the definition of each technical term.**

1. **Please describe the relationship between UML and methods.**
2. **Why do we define the use case model?**
3. **What is a use case description ?**
4. **What is an collaboration of UML?**
5. **What are analysis (or problem domain) classes?**
6. **What are design classes?**
7. **How can we define the interaction among objects using UML notations?**
8. **How can we define the behavior (or lifecycle) of an object using UML notations?**
9. **What is a stereotype of UML?**