Outline of UML and Unified Process

Koichiro OCHIMIZU

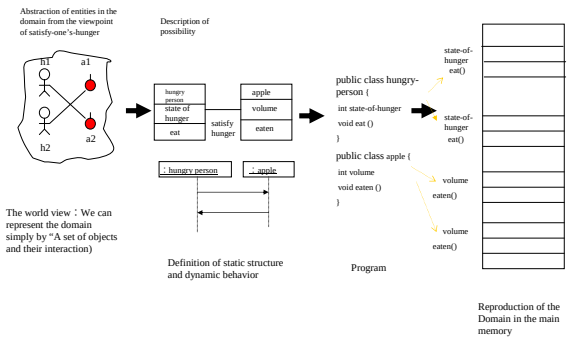
School of Information Science

JAIST

Schedule

- Feb. 27th
  - 13:00 Scope and Goal
  - 14:30 Basic Concepts on Representing the World (object, class, association, …)
- Feb. 28th
  - 13:00 Basic Concepts on Interaction (message passing, operation, method, polymorphism)
  - 14:30 Basic Concepts on Reuse (super class, class inheritance, interface inheritance)
- March 1st
  - 13:00 Introduction to Java Programming
  - 14:30 Outline of UML and Unified Process

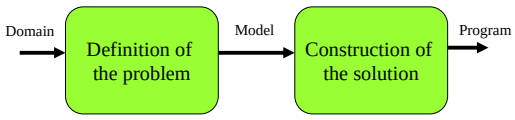
Modeling the Domain



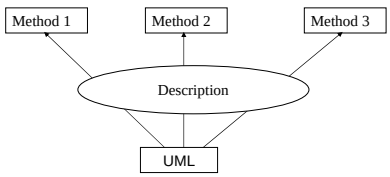
Object Oriented Analysis/Design/Programming

Iterative and Incremental

OOA ***OOD/OOP***



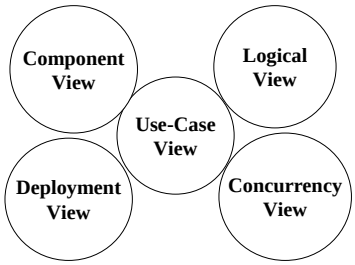
Relationship between Methods and UML



UML1.5

- Very popular now and help us make and analyze:
  - Use-case Diagrams for defining functional requirements
  - Collaboration 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

The Views in UML



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

Relationships between views and diagrams in UML

- Use-Case View
  - Use-Case Diagram
- Logical View
  - Class Diagram, Object Diagram
  - State Diagram, Sequence Diagram, Collaboration Diagram, Activity Diagram
- Concurrency View
  - State Diagram, Sequence Diagram, Collaboration Diagram, Activity Diagram
  - Component Diagram,, Deployment Diagram
- Deployment View
  - Deployment Diagram
- Component View
  - Component Diagram

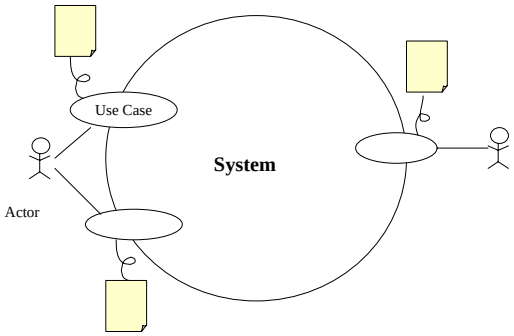
The Unified Software Development Process

- Use-Case Model
  - Use-Case Diagram
- Analysis Model
  - describe "Realization of a Use-Case" by a Collaboration Diagram and a Flow of Event Description
- Design Model
  - Class Diagram , Sequence Diagram, and Statechart Diagram
- Deployment Model
  - Deployment Diagram
- Implementation Model
  - Component Diagram
- Test Model
  - Test Case

Use Case Description

Event Sequences between actors and the system

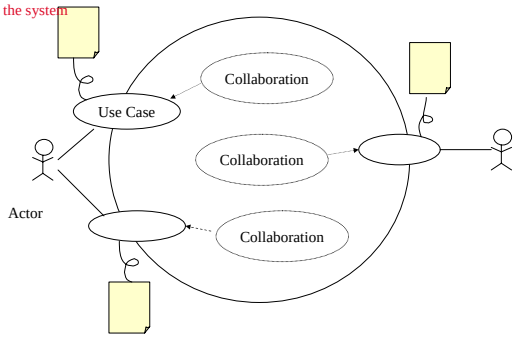
Functional Requirements



Use Case Description

Event Sequences between actors and the system

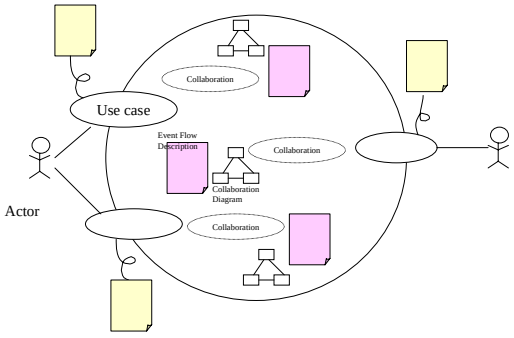
Analysis of inside of the system



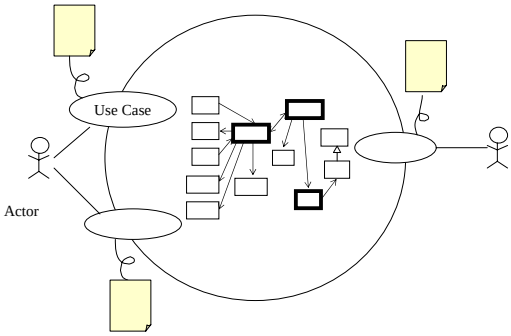
Use Case Description

Event Sequences between actors and the System

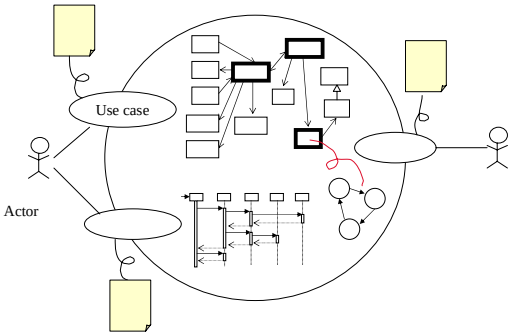
Analysis Classes



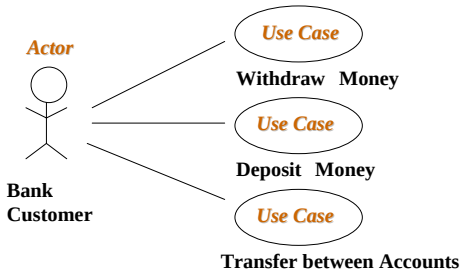
Class Diagram (Analysis Class + Design Class)



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



A Use-Case diagram with an actor and three use cases



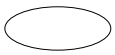
I. Jacobson, G.Booch, J.Rumbaugh,"The Unified Software Development Process", Addison Wesley, 1999.

UML Notations

Actor: icon for Stereotype Actor

Bank Customer

Use Case



Withdraw Money

Stereotypes

Stereotype/keyword	Applies to symbol	Meaning
Actor 	class	Specifies a coherent set of roles that users of use cases play when interacting these use cases

G.Booch, J.Rumbaugh, I. Jacobson , "The Unified Modeling Language User Guide", Addison Wesley, 1999.

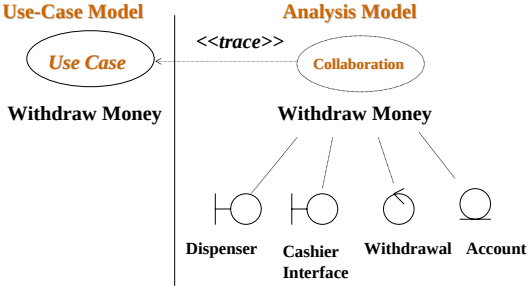
The Withdraw Money Use Case Description

1. **The Bank Customer identifies himself or herself**
2. **The Bank Customer chooses from which account to withdraw money and specifies how much to withdraw**
3. **The system decreases the amount from the account and dispenses the money.**

Use case are also used as "placeholders" for nonfuctional requirements

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

The Realization of a Use Case in the Analysis Model



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

UML notation

collaboration name — **Collaboration**

A collaboration gives a name to the mechanism of the system

It also serve as the realization of a use case

It defines a group of objects and their interaction

<<*trace*>> — A directed dashed line means dependency. In this case, trace dependency

G.Booch, J.Rumbaugh, I. Jacobson , "The Unified Modeling Language User Guide", Addison Wesley, 1999.

Analysis Stereotypes

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

Boundary — Dispenser, Cashier Interface

Control — Withdrawal

Entity — Account

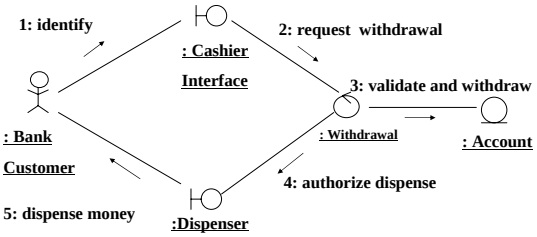
I. Jacobson, G.Booch, J.Rumbaugh,"The Unified Software Development Process", Addison Wesley, 1999.

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.

A collaboration diagram for the Withdraw Money use-case realization in the analysis model



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

UML notation

1: identify — **Message**

G.Booch, J.Rumbaugh, I. Jacobson , "The Unified Modeling Language User Guide", Addison Wesley, 1999.

Flow of Events Description of a Use-Case Realization

A **Bank Customer** chooses to withdraw money and activates the **Cashier Interface** object. The Bank Customer identifies himself or herself and specifies how much to withdraw and from which (1).

The **Cashier Interface** verifies the Bank Customer's identity and asks a Withdrawal object to perform the transaction (2).

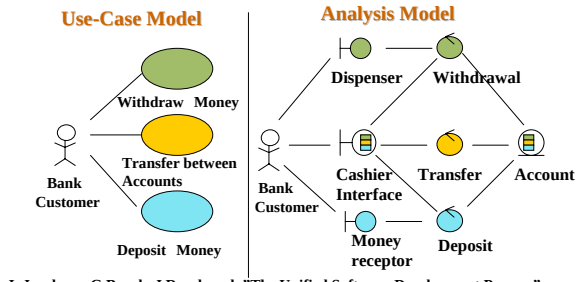
If the **Bank Customer's** identity is valid, the **Withdrawal** object is asked to confirm that the bank customer has the right to withdraw the specified amount from the **Account**. The **Withdrawal** object confirms this by asking the **Account** object to validate the request and, if the request Is valid, withdraw the amount (3) .

Then the **Withdrawal** object authorizes the **Dispenser** to dispense the amount that the **Bank Customer** requested (4) .

The **Bank Customer** then receives the requested amount of money (5) .

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

A Class Participating in Several Use-Case Realizations in Analysis Model



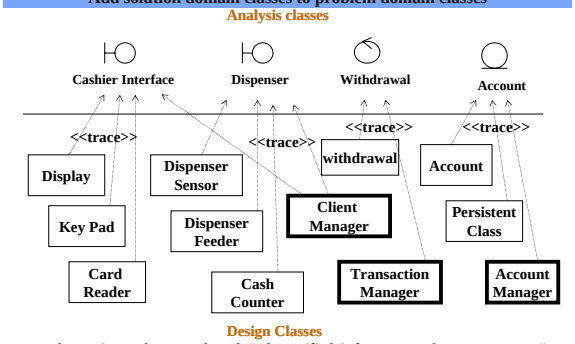
I. Jacobson, G.Booch, J.Rumbaugh,"The Unified Software Development Process", Addison Wesley, 1999.

Analysis Class

- Focus on functional requirements in defining analysis class. Deal with non functional requirements in design phase or implementation phase.
- Make the class responsibility clear
- Define attributes that exist in a real world
- Define association but do not Include details like navigation
- Use stereotype classes; <<boundary>>, <<control>>, and <<entity>>.

Analysis Class and Design Class

Add solution domain classes to problem domain classes



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

UML notation

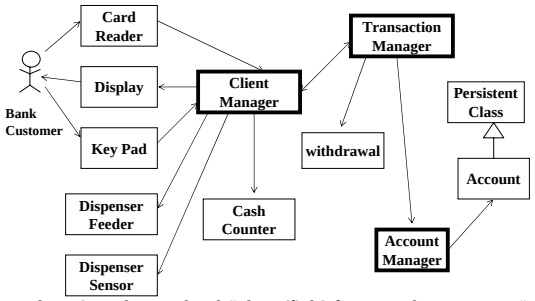
Active Class

Has a thread and can initiate a control activity

G.Booch, J.Rumbaugh, I. Jacobson , "The Unified Modeling Language User Guide", Addison Wesley, 1999.

Class Diagram

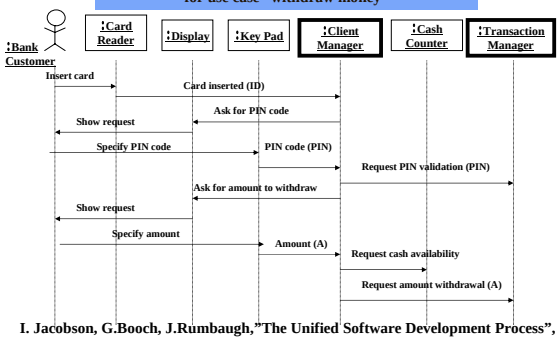
for use case "withdraw money"



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

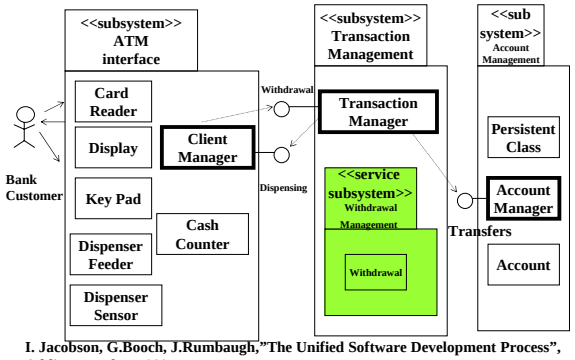
Sequence Diagram

for use case "withdraw money"



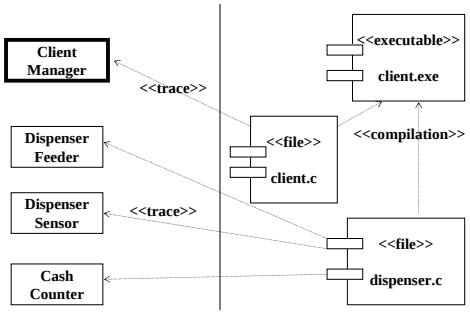
I. Jacobson, G.Booch, J.Rumbaugh,"The Unified Software Development Process", Addison Wesley, 1999.

Group Classes into Subsystems



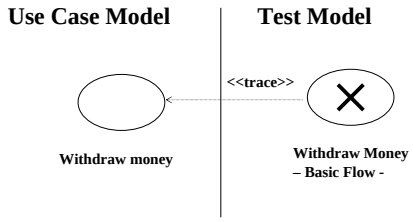
I. Jacobson, G.Booch, J.Rumbaugh,"The Unified Software Development Process", Addison Wesley, 1999.

Components that implements Design Classes



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

Test Cases are derived from Use Case



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

What do Software Engineering Projects consider important? by Pete McBreen

- **Traditional Waterfall Projects**
  - Specialization of staff into different roles to support the different phases is claimed to promote efficiency by reducing the number of skills a person needs.
  - With clear milestones between phases and known dependencies between deliverables, it is easy to display a waterfall project on a PERT chart.
  - Comprehensive documentation is important, so that at the end of the project it is possible to justify the overall costs. This supports the tracking of the project because it makes everything available for external review. A side benefit of all of this documentation is traceability.
- **Unified Process (supports Incremental development in the context of a phased approach)**
  - Inception(evaluating the economic feasibility of the project, forcing the team to define the overall project scope, plan the remaining phases, and produce estimates)
  - Elaboration (evaluating the technical feasibility of the project by creating and validating the overall software architecture)
  - Construction (at the end of each increment, new and changed requirements can be incorporated into the plans, and the estimates can be refined based on experiences in the previous increments)

Pete McBreen, "Questining eXtreme Programming", Addison-Wesley, 2003.

What do Software Engineering Projects consider important? by Pete McBreen

- **Open Source Project**
  - Free, unrestricted access to the source code so that developers can share and learn
  - The reputation of the project's developers
  - Frequent releases back to the community
- **Agile**
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan
- **XP**
  - A predictable, sustained, and sustainable pace in the face of changing requirements
  - A collaborative, supportive environment for developers
  - Enhancement of the skills and knowledge of the development team

Pete McBreen, "Questining eXtreme Programming", Addison-Wesley, 2003.

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?**