

Outline of Unified Process

Koichiro OCHIMIZU

School of Information Science

JAIST

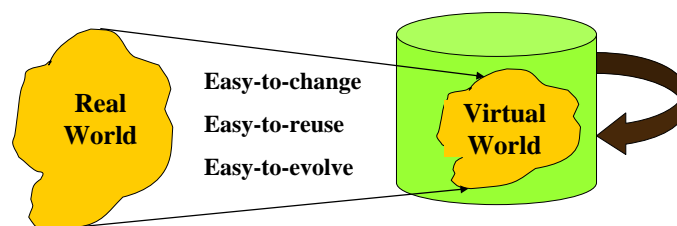
Schedule(3/3)

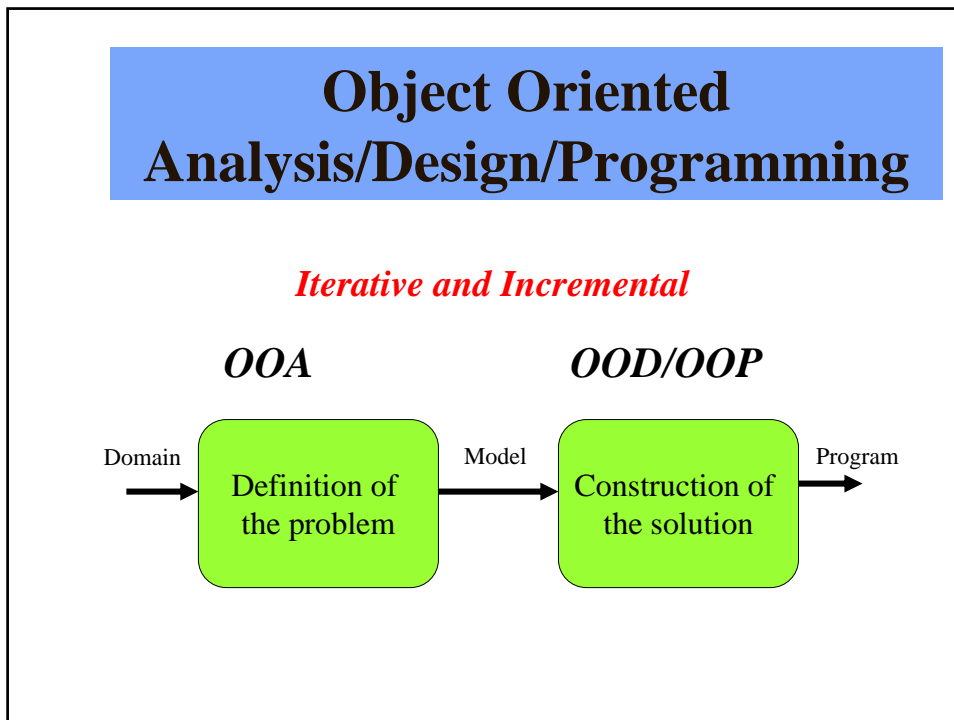
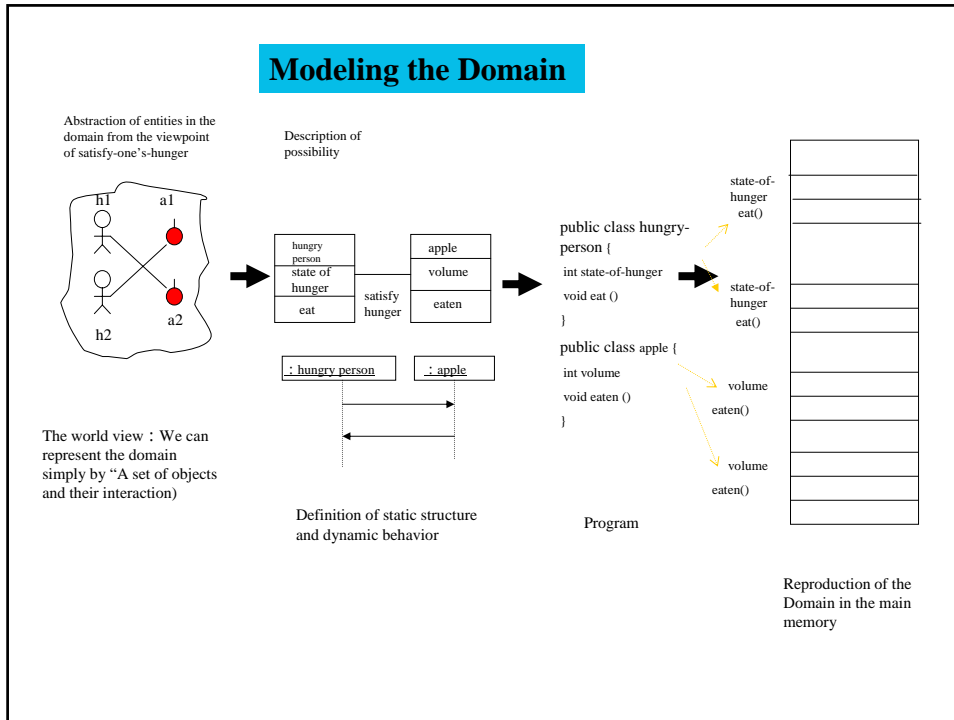
- March 12
 - 13:00 **Unified Process and COMET**
 - 14:30 Case Study of Elevator Control System
(problem definition, use case model)
- March 13
 - 13:00 Case Study of Elevator Control System
(finding analysis classes by developing a consolidated communication diagram)
 - 14:30 Case Study of Elevator Control System
(sub-system structuring and task structuring)
- March 14
 - 13:00 Case Study of Elevator Control System
(performance analysis)
 - 14:30 UML2.0 and MDA

What is OOA/OOD/OOP approach ?

How to incorporate three major merits of OOT into a system structure through OOSD

- *Project the real world into the computer as you recognize and understand it.*
- *Maintain the virtual world constantly corresponding to mismatches between the real world and the virtual world and evolution of the real world.*



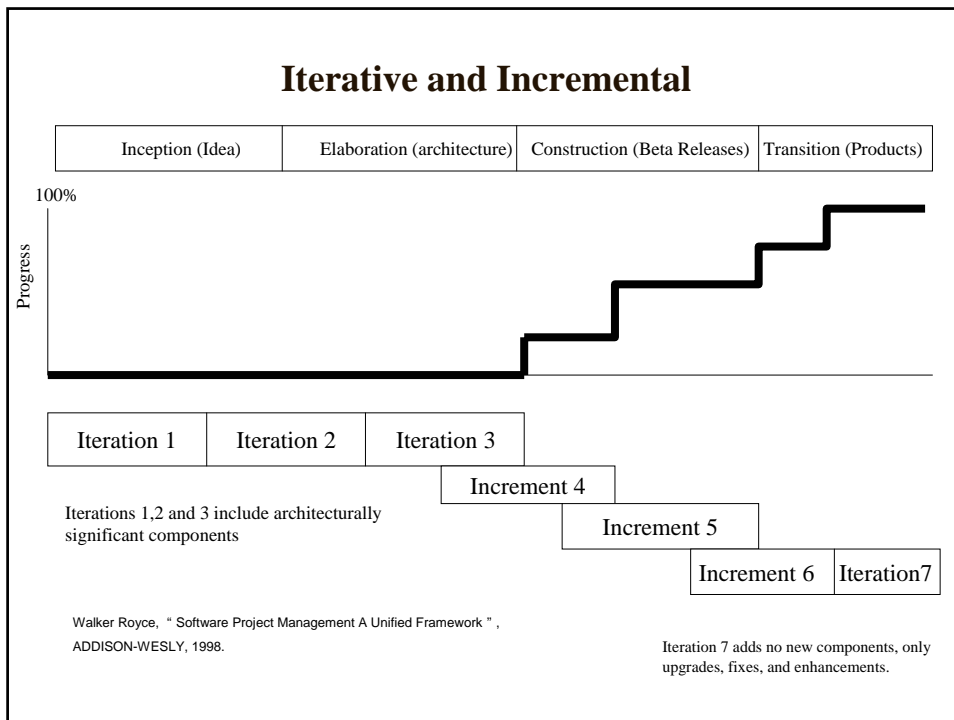
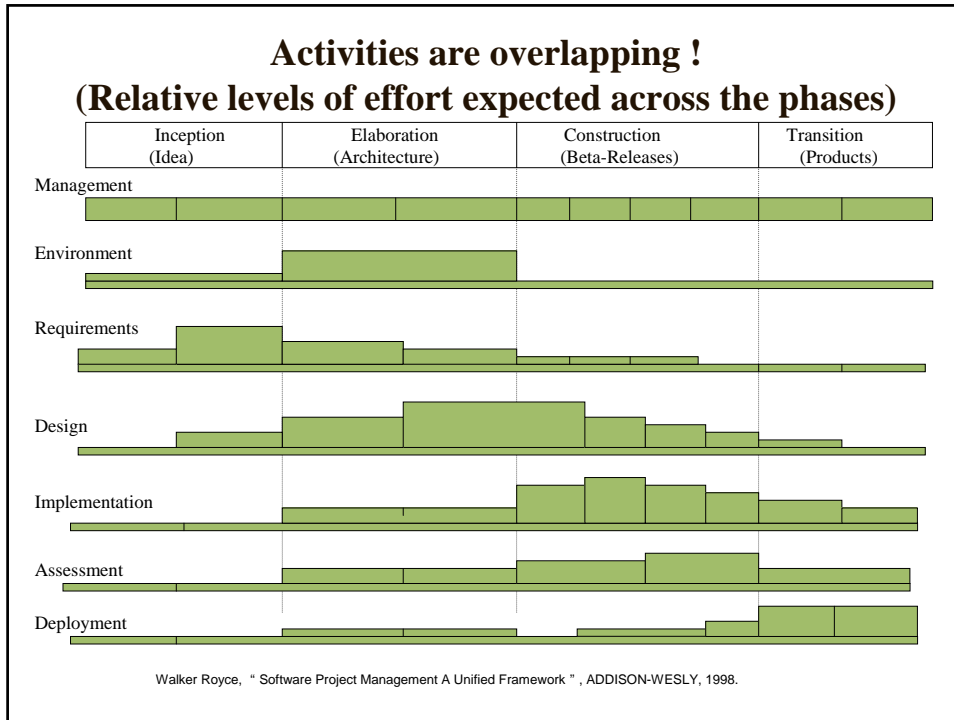


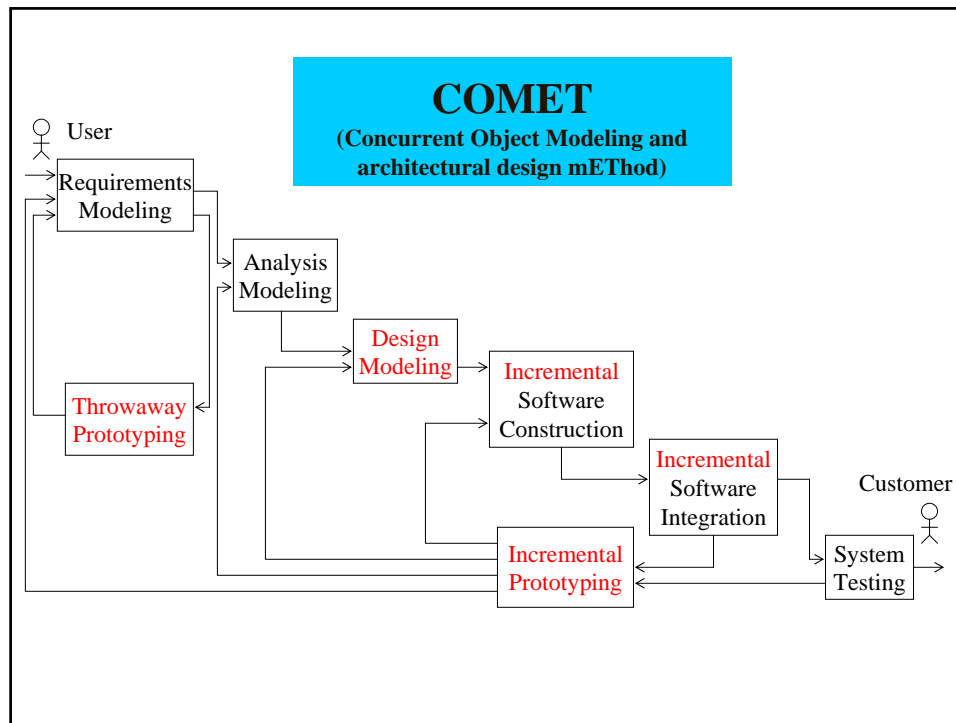
What is the Unified Process ?

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.



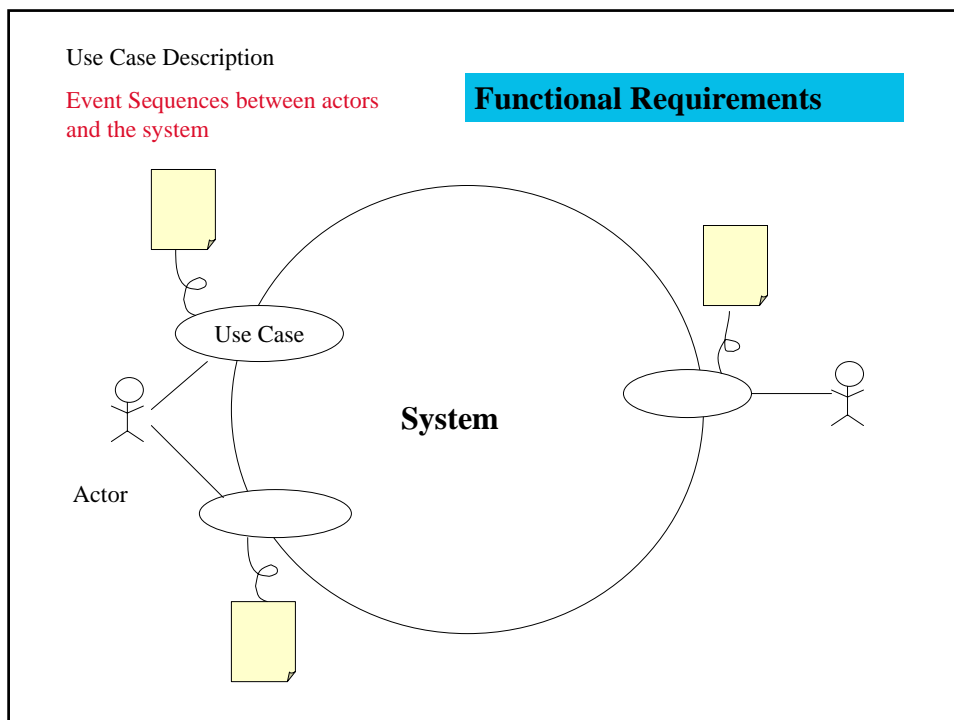


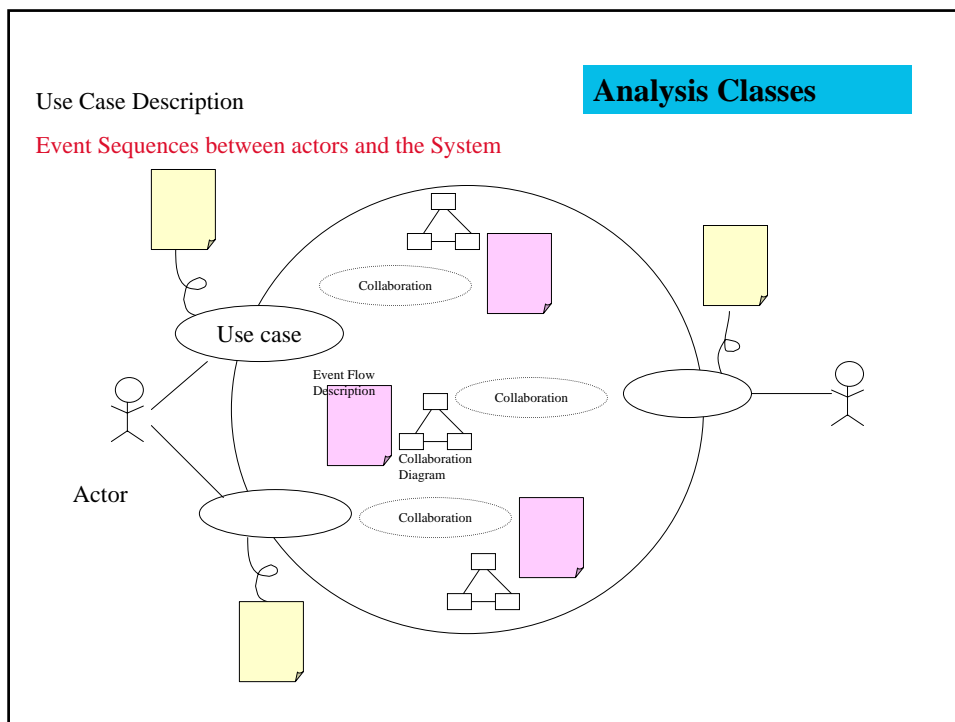
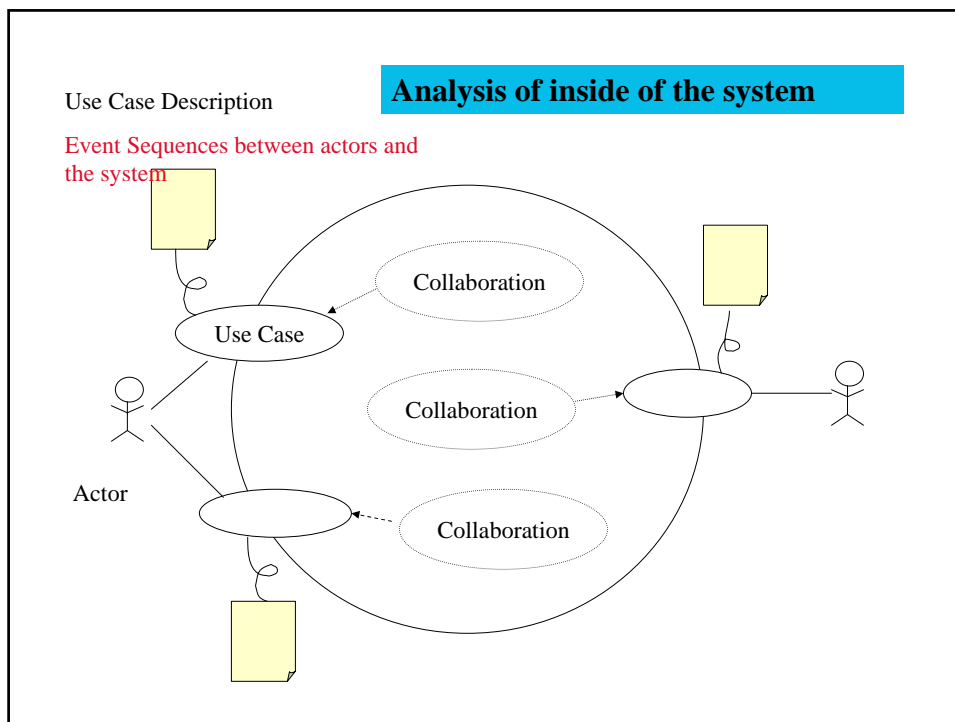
Backtracking to Iteration

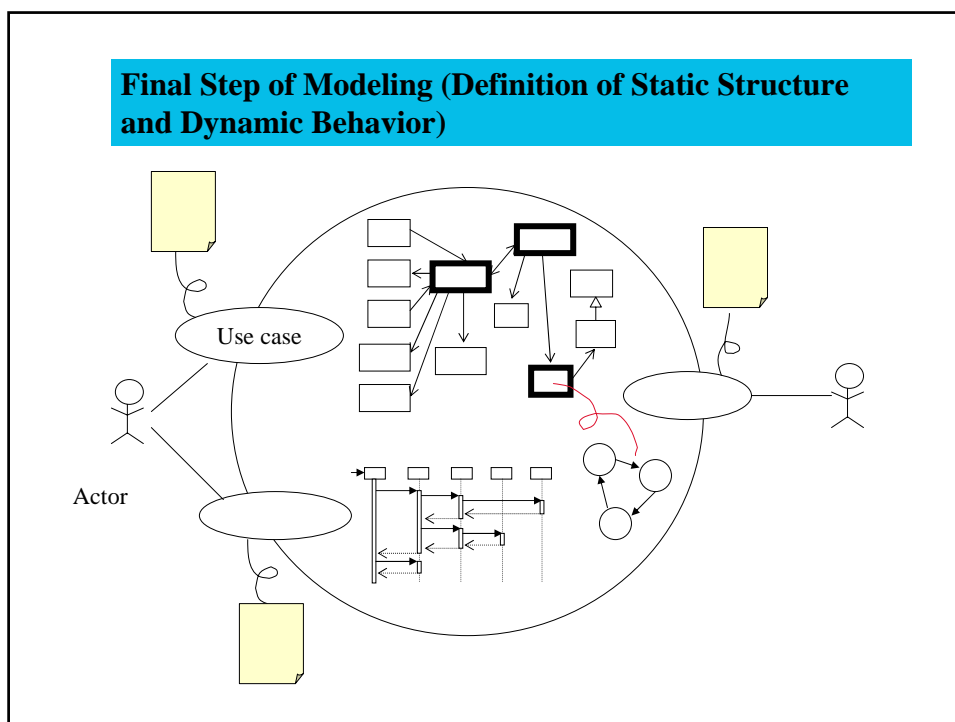
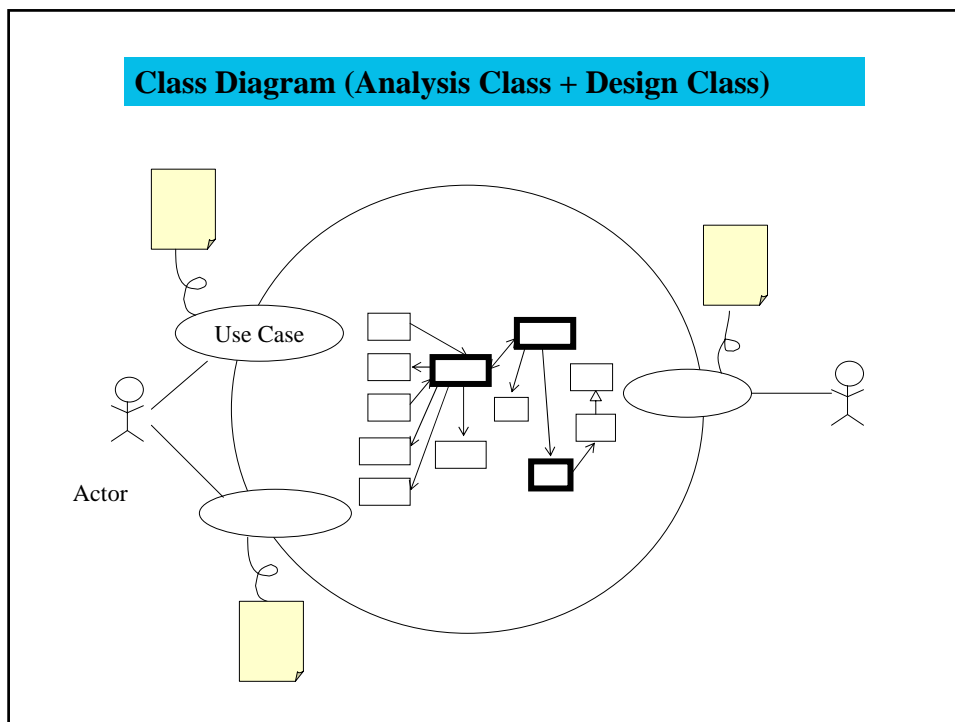
- Mini Waterfall Model, Spiral Model
 - Enable us to detect and deal with risks on Quality, Cost, and Delivery by Iteration in early phases.
- Prototyping
 - Try to elicitate the real requirements by including users into Iteration
- Iterative and Incremental Model
 - Find project-specific risks in early phases in software development process by iteration. Improve the process at each increment.

Usecase-driven approach

- define functional requirements
- find analysis classes
- design the static structure
- define objects interaction
- define the behavior of each object
- allocate objects to machines

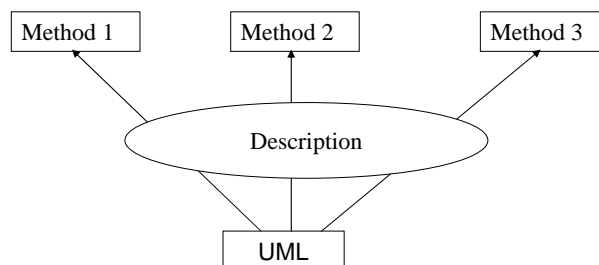




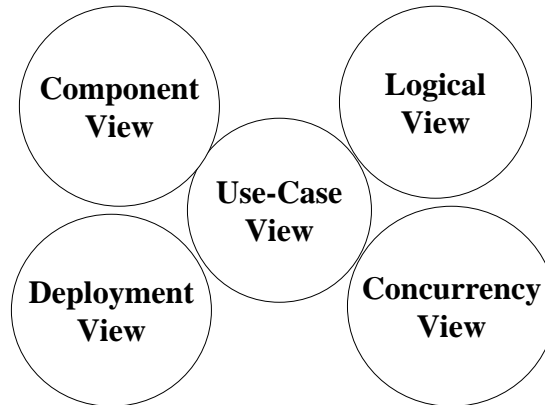


Role of UML

Relationship between Methods and UML



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

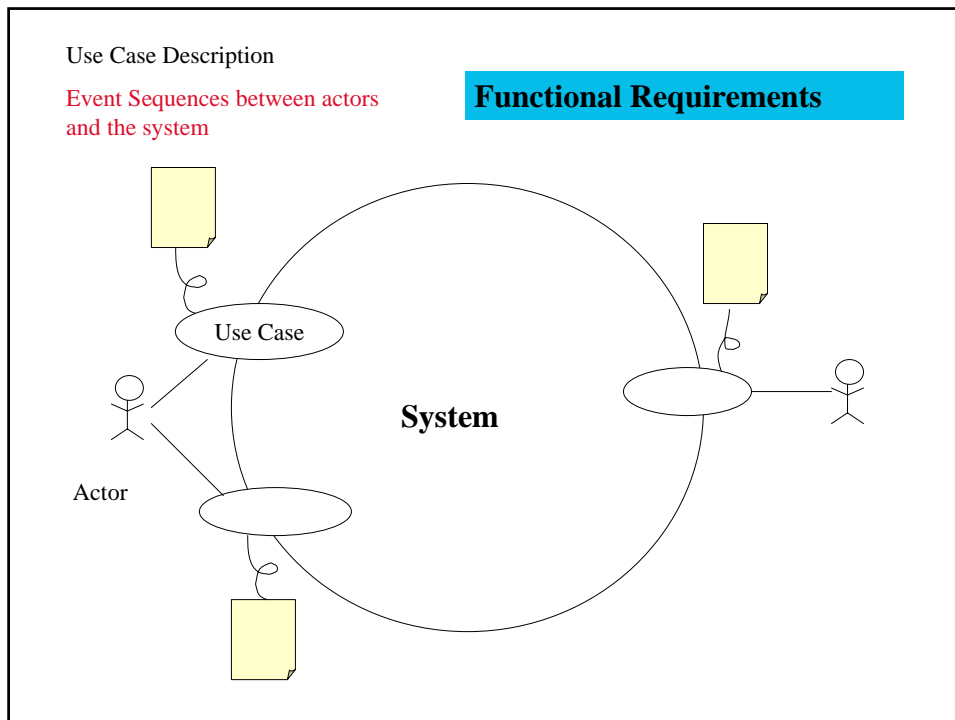
The Unified Software Development Process

- 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

Usecase Driven process and 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

ATM example



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.

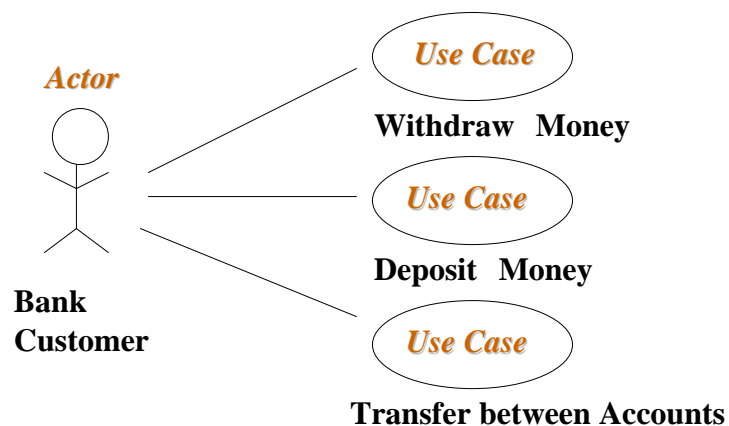
H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

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.

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.

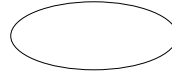
UML Notations

Actor: icon for Stereotype Actor



Bank Customer

Use Case




Withdraw Money

Stereotypes

A stereotype extension mechanism defines a new kind of model element based on an existing model element with some extra semantics that are not present in the existing one.

Stereotype/keyword Applies to symbol

Meaning

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

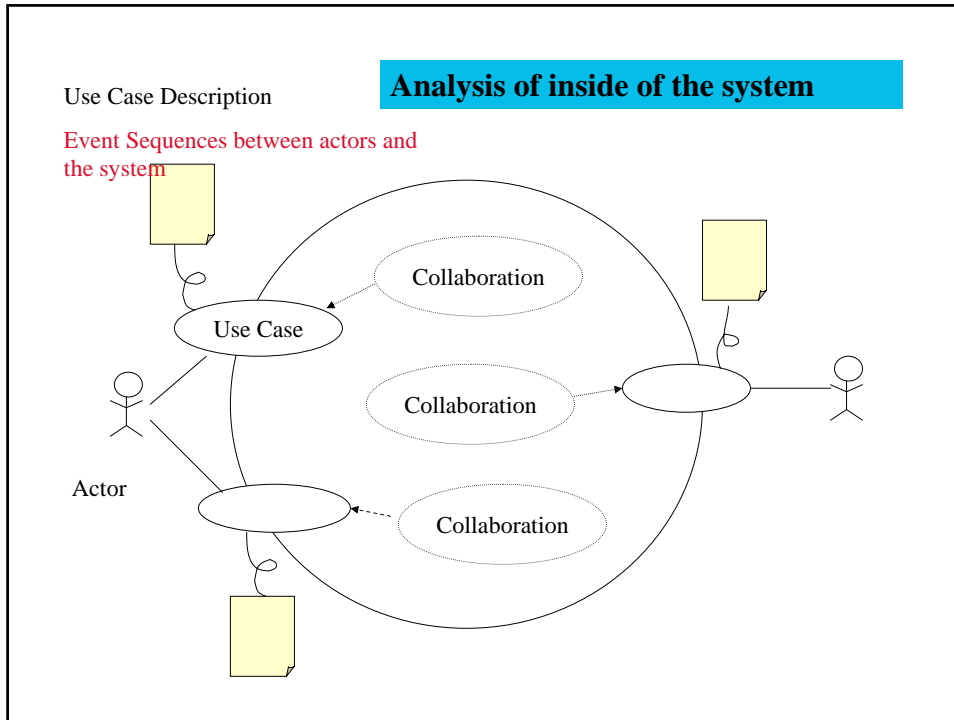
G.Booch, J.Rumbaugh, I. Jacobson , "The Unified Modeling Language User Guide", Addison Wesley, 1999.

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 nonfunctional requirements

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

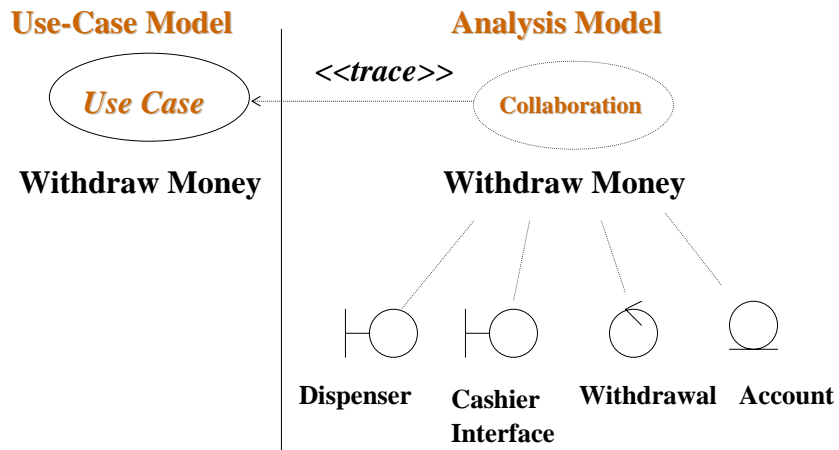
A directed dashed line means dependency

`<<trace>>` In this case, trace dependency

←

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

The Realization of a Use Case in the Analysis Model

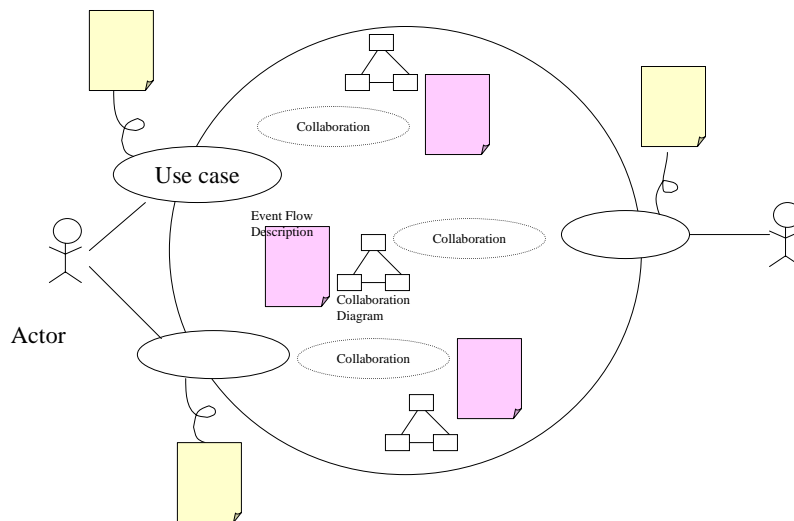


I. Jacobson, G.Booch, J.Rumbaugh, "The Unified Software Development Process", Addison Wesley, 1999.

Use Case Description

Analysis Classes

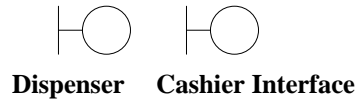
Event Sequences between actors and the System



Analysis Stereotypes

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

Boundary



Control



Entity



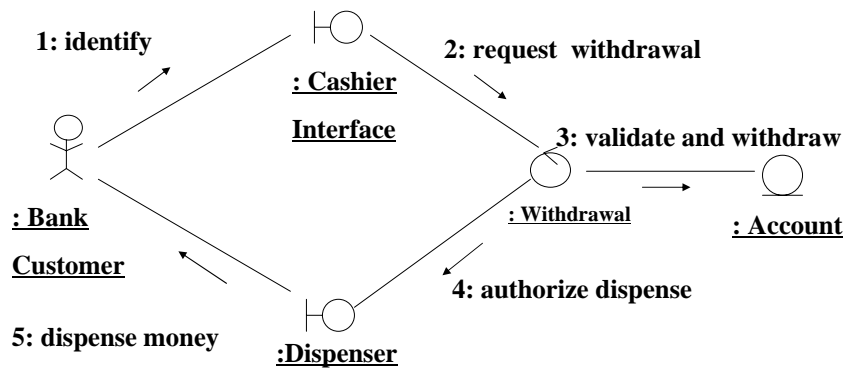
I. Jacobson, G.Booch, J.Rumbaugh,"The Unified Software Development Process", Addison Wesley, 1999.

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.

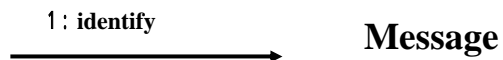
I. Jacobson, G.Booch, J.Rumbaugh,"The Unified Software Development Process", Addison Wesley, 1999.

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



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).

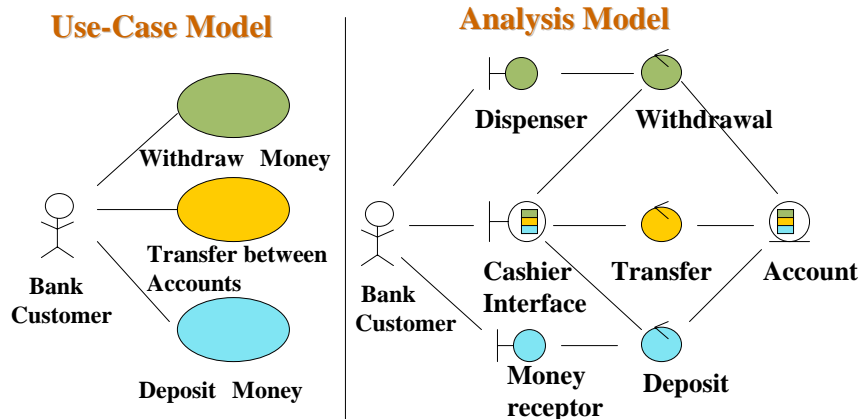
If the **Bank Customer's** identity is valid, the **Withdrawal** object is asked to confirm that the bank customer has the right to withdraw the specified amount from the **Account**. The **Withdrawal** object confirms this by asking the **Account** object to validate the request and, if the request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

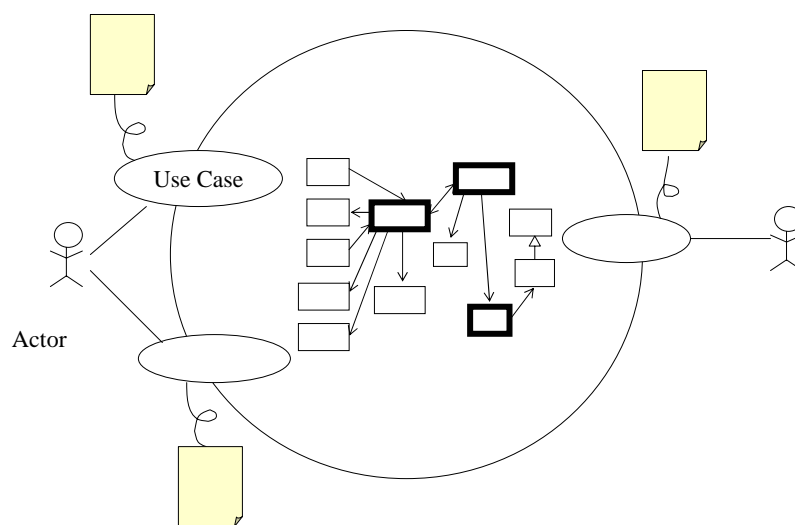


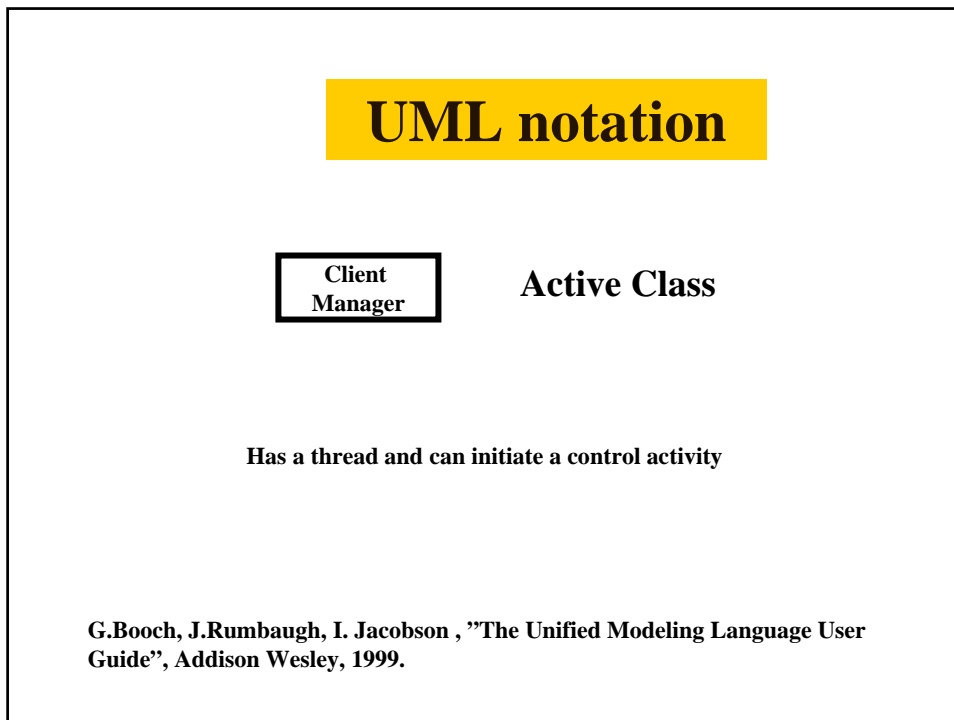
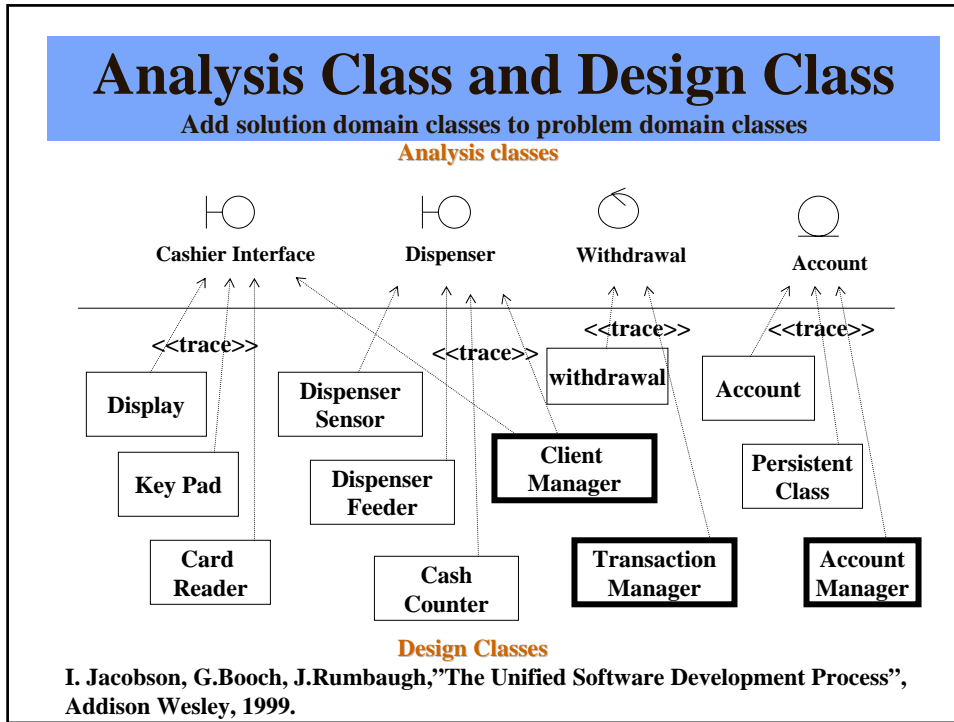
I. Jacobson, G.Booch, J.Rumbaugh,"The Unified Software Development Process", Addison Wesley, 1999.

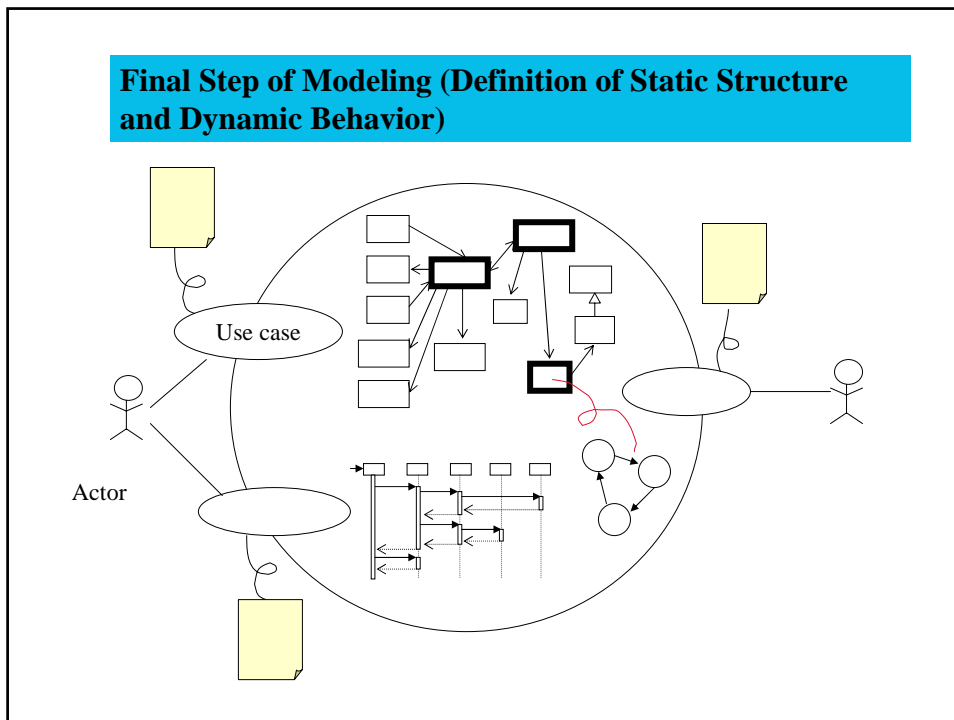
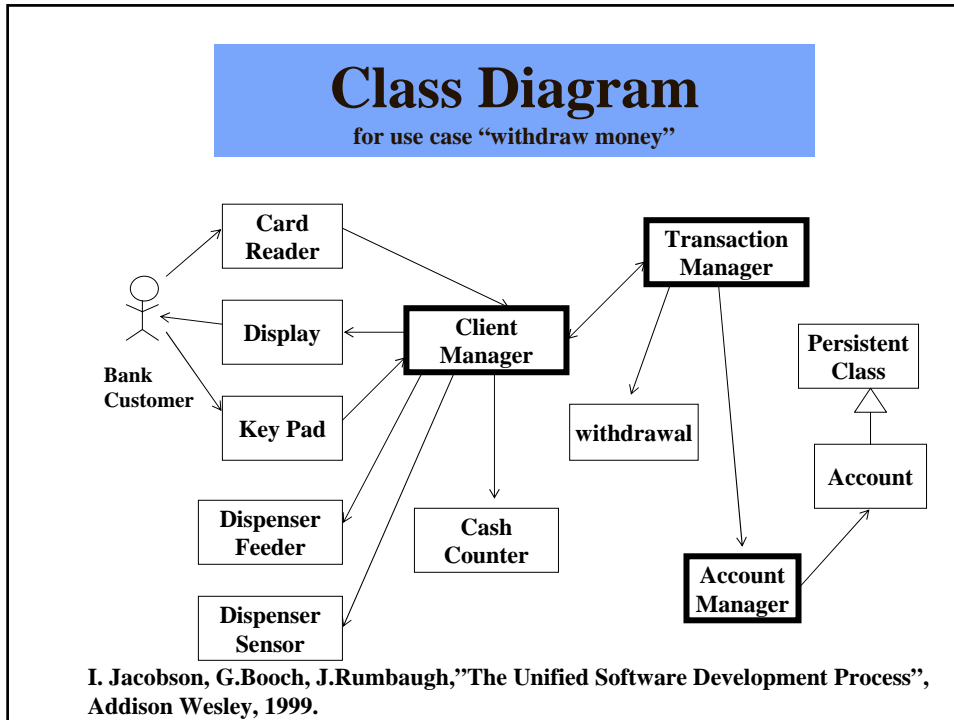
Analysis Class

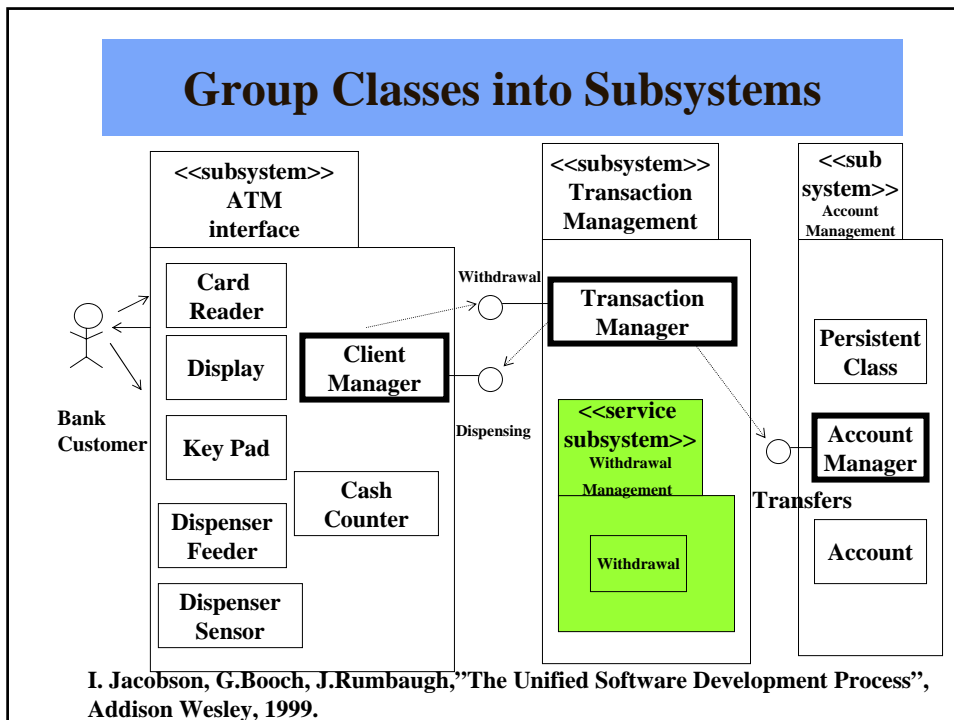
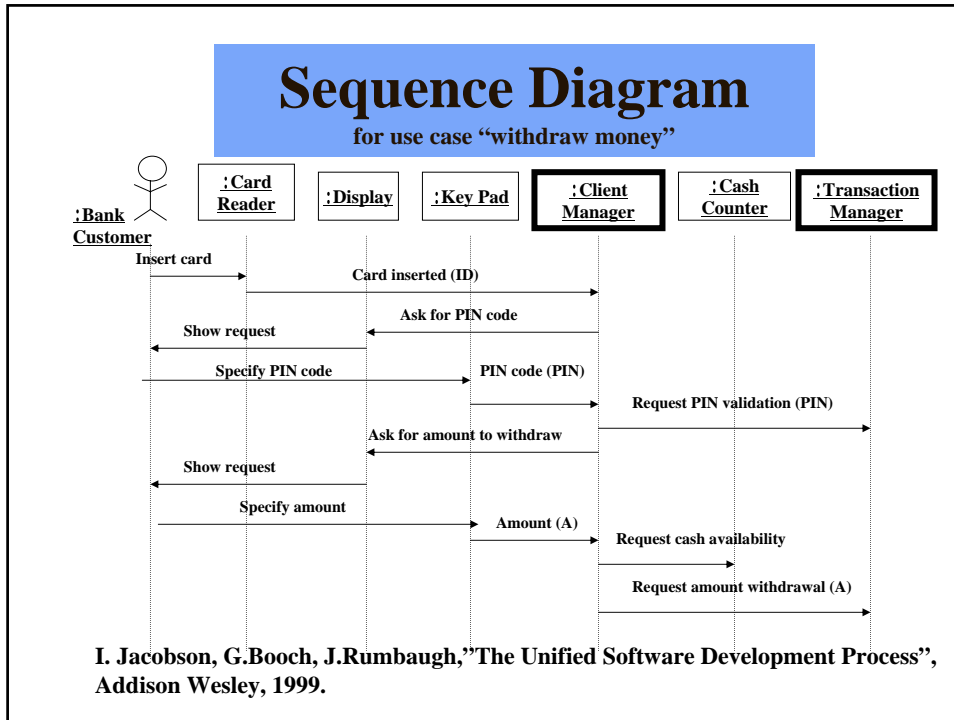
- Focus on functional requirements in defining analysis class. Deal with non functional requirements in design phase or implementation phase.
- Make the class responsibility clear
- Define attributes that exist in a real world
- Define association but do not Include details like navigation
- Use stereotype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

Class Diagram (Analysis Class + Design Class)

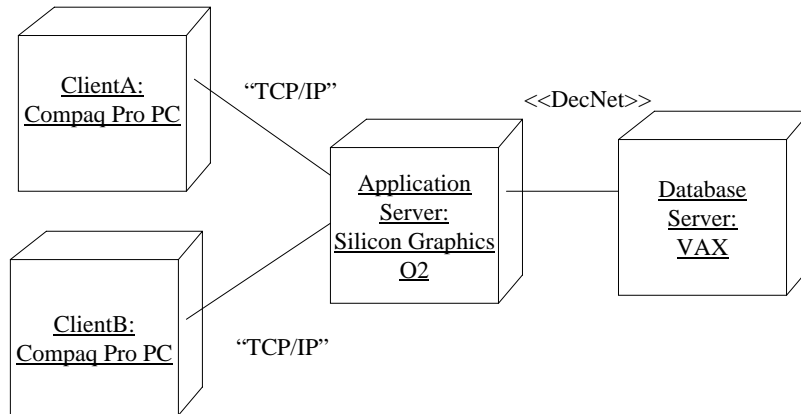






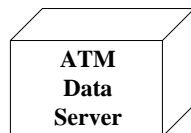


Communication Association between nodes



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

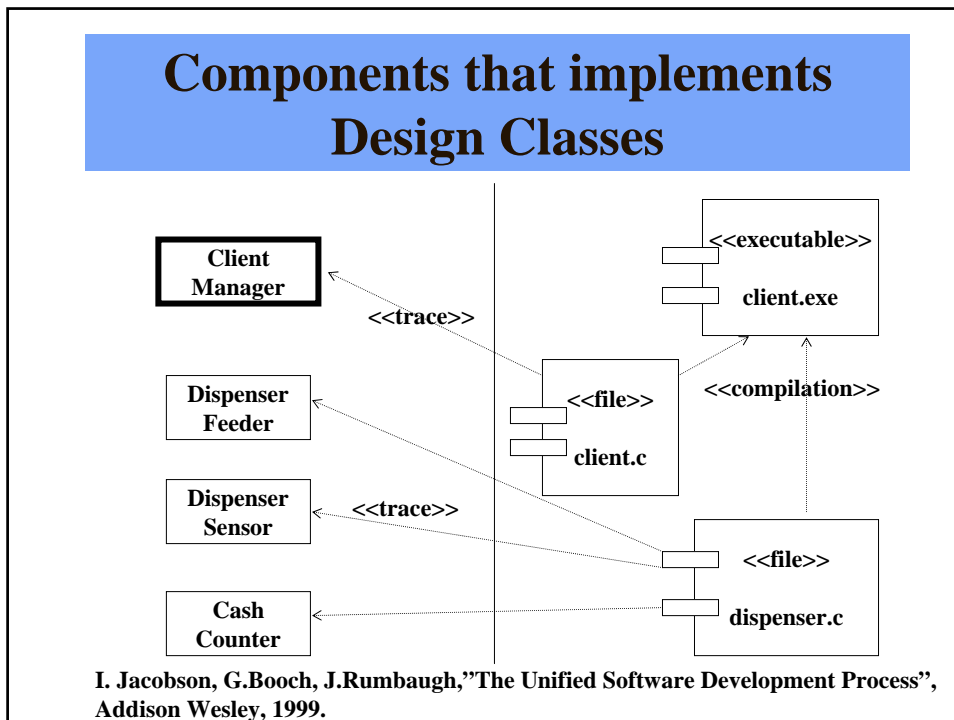
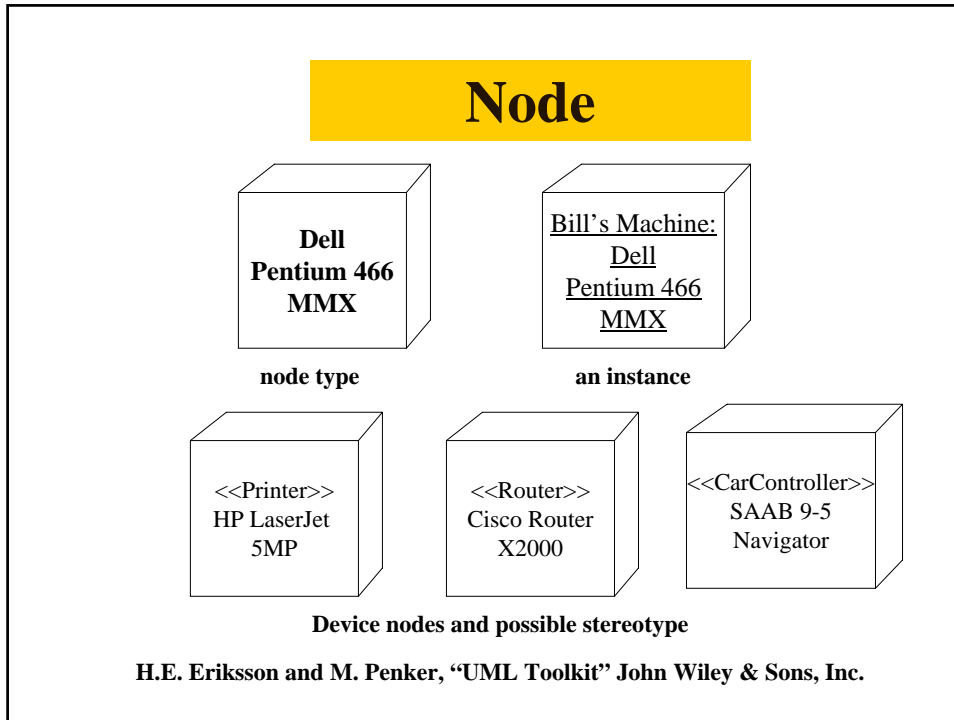
UML notation



Node

A physical element that exists at run time and that represents a computational resource, generally having at least some memory and, often times, processing capability

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



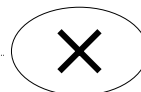
Test Cases are derived from Use Case

Use Case Model



Withdraw money

Test Model



Withdraw Money
- Basic Flow -

<<trace>>

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

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?