

Elevator Control System

Koichiro Ochimizu
School of Information Science
Japan Advanced Institute of Science and Technology

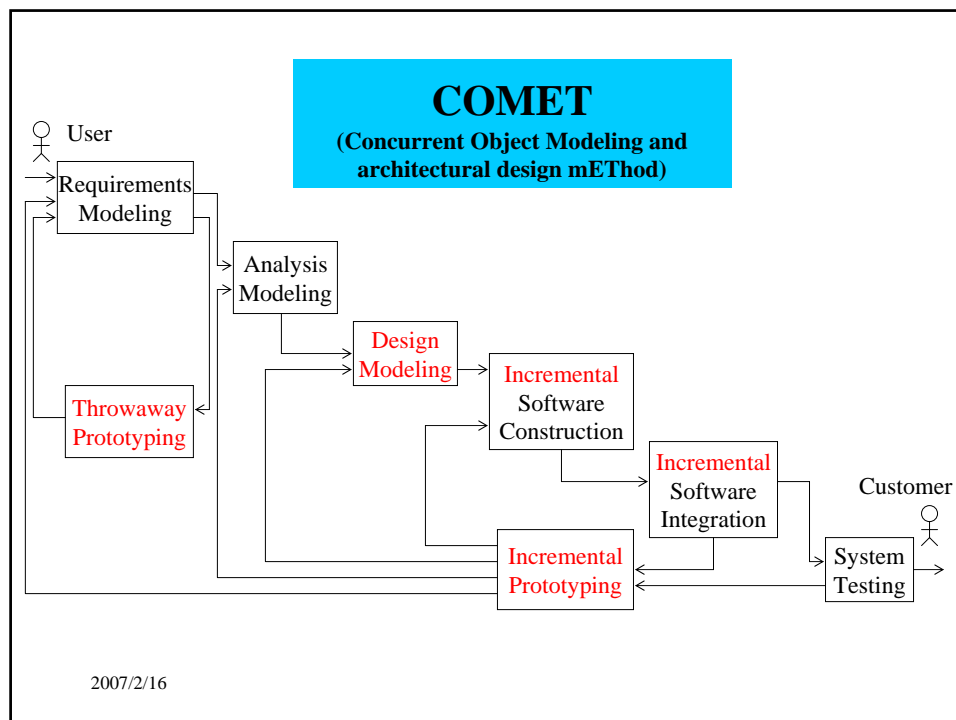
Schedule(3/3)

- 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

Exercises

- *Elevator Control System*
- *Banking System*
- *Cruise Control and Monitoring System*
- *Distributed Factory Automation System*
- *Electronic Commerce System*

Hassan Gomaa, “Designing Concurrent, Distributed, and Real-Time Applications with UML”, Addison-Wesley, Object Technology Series, 2000.



Activities of each Phase(1/2)

- **Requirements Modeling** functional requirements defined by **actors** and **use cases**
- **Analysis Modeling**
 - Static model: structural relationships among problem domain classes depicted on **class diagrams**,
 - Dynamic model: objects and their interactions depicted on either **collaboration diagrams** or **sequence diagrams**.
 - The state-dependent aspects of the system are defined using hierarchical statecharts (**finite state machines**).
- **Design Modeling** The software architecture of the system is designed. The analysis model is mapped to an operational environment. **Subsystem** structuring criteria are provided.

Activities of each Phase(2/2)

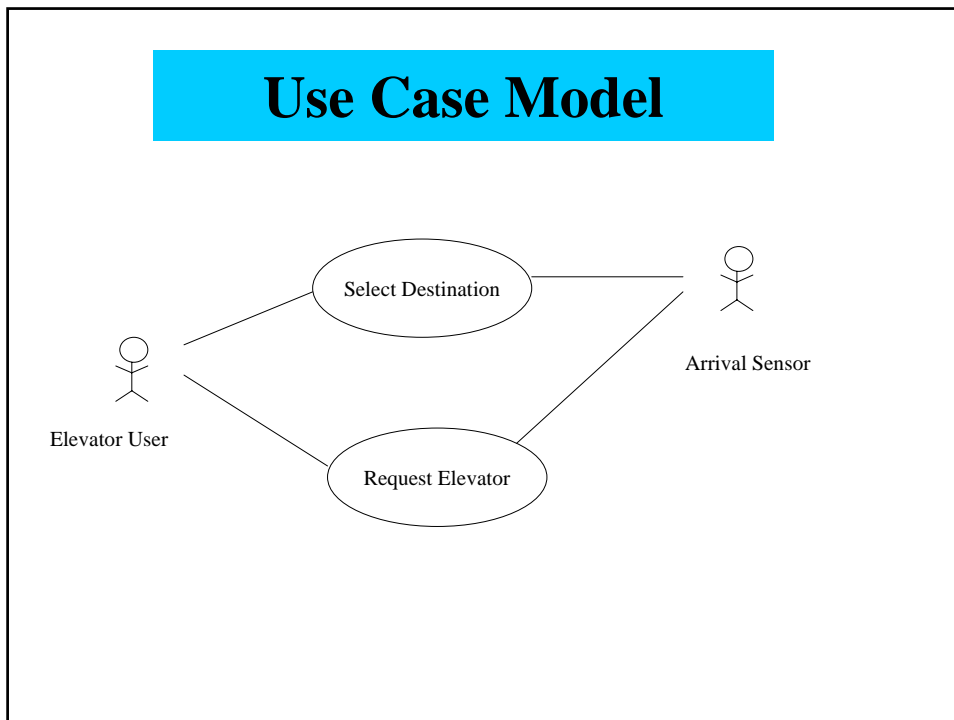
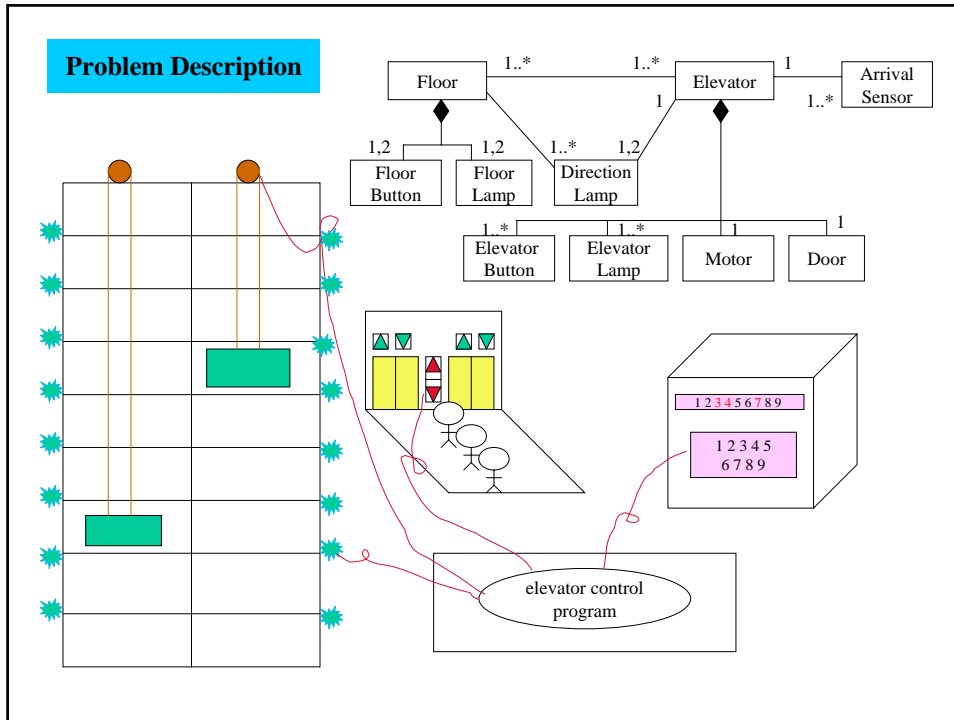
- **Incremental Software Construction** Selecting a subset of the system to be constructed for each increment. The subset is determined by choosing the use cases to be Included in this increment. Incremental software construction consists of the detailed design, coding, and unit testing of the classes in the subset.
- **Incremental Software Integration** the integration testing of each increment is performed. Integration test cases are developed for each use case. Interfaces between the objects that participates in each use case are tested.
- **System Testing** testing the system against Its functional requirements.

COMET Process

- Problem Definition
 - Problem Description
 - Definition of Functional Requirements by a Use Case Diagram and Use Case descriptions
- Analysis
 - Static Modeling of objects in a problem domain.
 - Dynamic Modeling in a problem domain
 - Collaboration Diagrams
 - State Charts
 - Consolidated State Charts
 - Consolidated Collaboration Diagram
- Architecture Design (Distributed, Non Distributed)
 - Subsystem Design
 - Task Design
 - Information Hiding Class Design
- Detailed Design
 - Connectors Design
 - Composite Task Design
 - Deployment
- Performance Analysis
 - Worst-case scenario analysis
- Programming (Incremental)

Problem Description

- Definition of Functional Requirements by Use Cases
- Describing a class diagram of real world entities is recommended.

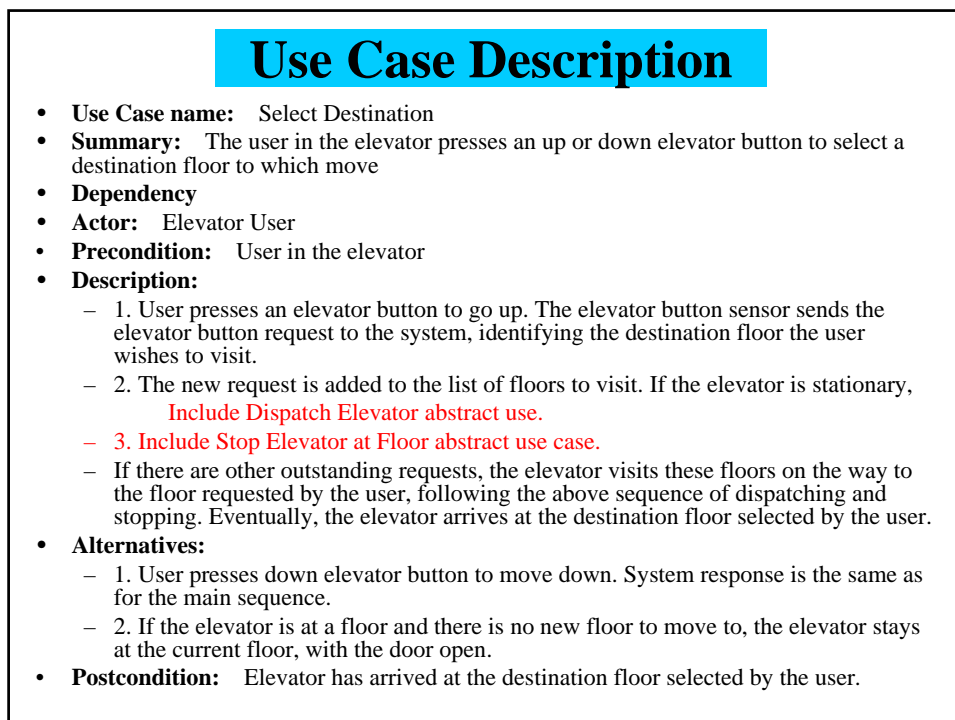
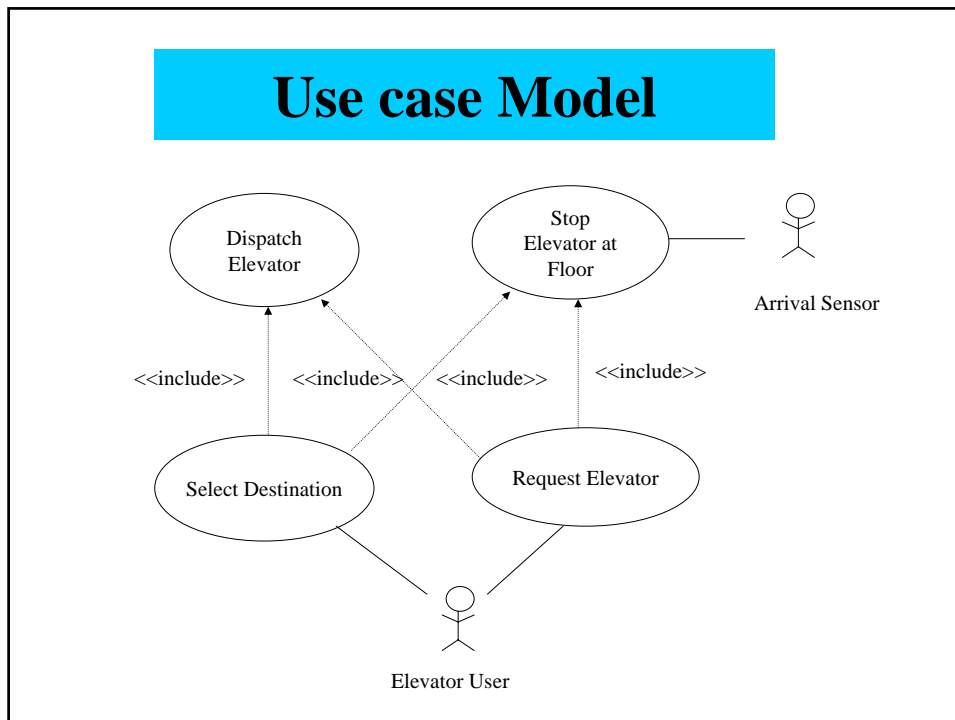


Use Case Description

- **Use Case name:** Select Destination
- **Summary:** The user in the elevator presses an up or down elevator button to select a destination floor to which move
- **Dependency**
- **Actor:** Elevator User (primary), Arrival Sensor
- **Precondition:** User in the elevator
- **Description:**
 - 1. User presses an elevator button to go up. The elevator button sensor sends the elevator button request to the system, identifying the destination floor the user wishes to visit.
 - 2. The new request is added to the list of floors to visit. If the elevator is stationary, **The system determines in which direction the system should move in order to service the next request. The system commands the elevator door to close. When the door has closed, the system commands the motor to start moving the elevator, either up or down.**
 - 3. **As the elevator moves between floors, the arrival sensor detects that the elevator is approaching a floor and notifies the system. The system checks whether the elevator should stop at this floor. If so, the system commands the motor to stop. When the elevator has stopped, the system commands the elevator door to open.**
 - If there are other outstanding requests, the elevator visits these floors on the way to the floor requested by the user. Eventually, the elevator arrives at the destination floor selected by the user.
- **Alternatives:**
 - 1. User presses down elevator button to move down. System response is the same as for the main sequence.
 - 2. If the elevator is at a floor and there is no new floor to move to, the elevator stays at the current floor, with the door open.
- **Postcondition:** Elevator has arrived at the destination floor selected by the user.

Use Case Description

- **Use Case name:** Request Elevator
- **Summary:** The user at a floor presses an up or down floor button to request an elevator.
- **Dependency**
- **Actor:** Elevator User (primary), Arrival Sensor
- **Precondition:** User is at a floor and wants to an elevator
- **Description:**
 - 1. User presses an up floor button. The floor button sensor sends the user request to the system, identifying the floor number.
 - 2. The system selects an elevator to visit this floor. The new request is added to the list of floors to visit. If the elevator is stationary, **The system determines in which direction the system should move in order to service the next request. The system commands the elevator door to close. After the door has closed, the system commands the motor to start moving the elevator, either up or down.**
 - 3. **As the elevator moves between floors, the arrival sensor detects that the elevator is approaching a floor and notifies the system. The system checks whether the elevator should stop at this floor. If so, the system commands the motor to stop. When the elevator has stopped, the system commands the elevator door to open.**
 - If there are other outstanding requests, the elevator visits these floors on the way to the floor requested by the user. Eventually, the elevator arrives at the floor in response to the user request.
- **Alternatives:**
 - 1. User presses floor button to move down. System response is the same as for the main sequence.
 - 2. If the elevator is at a floor and there is no new floor to move to, the elevator stays at the current floor, with the door open.
- **Postcondition:** Elevator has arrived at the floor in response to user request.



Use Case Description

- **Use Case name:** Request Elevator
- **Summary:** The user at a floor presses an up or down floor button to request an elevator.
- **Dependency**
- **Actor:** Elevator User
- **Precondition:** User is at a floor and wants to an elevator
- **Description:**
 - 1. User presses an up floor button. The floor button sensor sends the user request to the system, identifying the floor number.
 - 2. The system selects an elevator to visit this floor. The new request is added to the list of floors to visit. If the elevator is stationary, **then include Dispatch Elevator abstract use case.**
 - 3. **Include Stop Elevator at Floor abstract use case.**
 - If there are other outstanding requests, the elevator visits these floors on the way to the floor requested by the user following the above sequence of dispatching and stopping. Eventually, the elevator arrives at the floor in response to the user request.
- **Alternatives:**
 - 1. User presses floor button to move down. System response is the same as for the main sequence.
 - 2. If the elevator is at a floor and there is no new floor to move to, the elevator stays at the current floor, with the door open.
- **Postcondition:** Elevator has arrived at the floor in response to user request.

Use Case Description

- **Use Case name:** Stop Elevator at Floor abstract Use Case
- **Summary:**
- **Dependency**
- **Actor:** Arrival Sensor
- **Precondition:** Elevator is moving
- **Description:**
 - **As the elevator moves between floors, the arrival sensor detects that the elevator is approaching a floor and notifies the system. The system checks whether the elevator should stop at this floor. If so, the system commands the motor to stop. When the elevator has stopped, the system commands the elevator door to open.**
- **Alternatives:**
 - The elevator is not required to stop at this floor and so continues past the floor.
- **Postcondition:** Elevator has stopped at floor, with door open.

Use Case Description

- **Use Case name:** Dispatch Elevator abstract use case
- **Summary:**
- **Dependency:**
- **Actor:**
- **Precondition:** Elevator is at a floor with the door open.
- **Description:**
 - The system determines in which direction the system should move in order to service the next request. The system commands the elevator door to close. After the door has closed, the system commands the motor to start moving the elevator, either up or down.
- **Alternatives:**
 - If the elevator is at a floor and there is no new floor to move to, the elevator stays at the current floor, with the door open.
- **Postcondition:** Elevator is moving in the commanded direction.