SWE 621:
Software Modeling and Architectural Design

Lecture Notes on Software Design

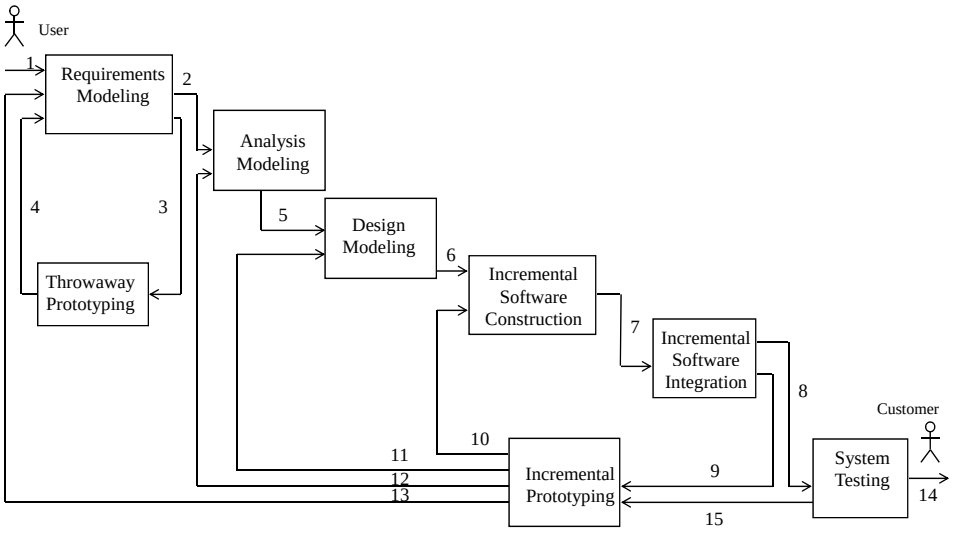
Lecture 14 - Course Review

Hassan Gomaa
Dept of Computer Science
George Mason University
Fairfax, VA

Overview

- Collaborative Object Modeling and architectural design mEThod (COMET)
  - Object Oriented Analysis and Design Method
  - Uses UML (Unified Modeling Language) notation
    - Standard approach for describing a software design
  - COMET = UML + Method
- Provides steps and guidelines for
  - Software Modeling and Design
  - From Use Case Models to Software Architecture
- H. Gomaa, *Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures*, Cambridge University Press, February 2011

Figure 5.1 COMET software life cycle model

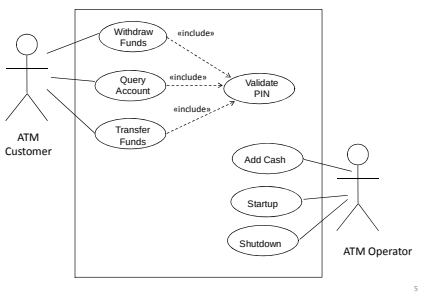


Object-Oriented Software Life Cycle

Requirements Modeling

- Requirements Modeling
  - Use Case Modeling
    - Define software functional requirements in terms of use cases and actors

Figure 21.1 Banking System use case model

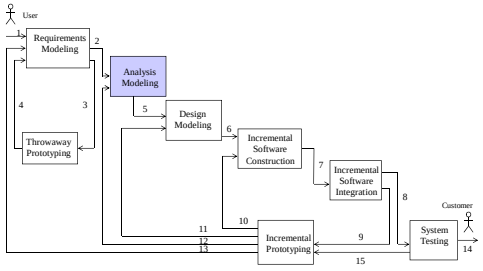


Object-Oriented Software Life Cycle

Analysis Modeling

- Analysis Modeling consists of
  - Static Modeling
  - Dynamic Modeling
    - State Machine modeling using statecharts
    - Object interaction modeling

Figure 5.1 COMET object-oriented software life cycle model

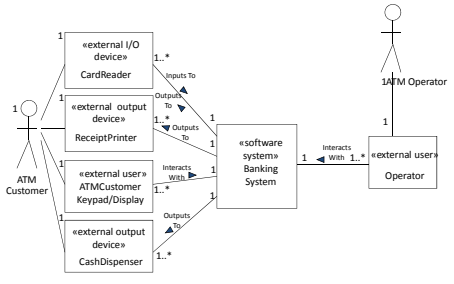


Object-Oriented Software Life Cycle

Analysis Modeling

- Static Modeling
  - Define structural relationships between classes
  - Depict classes and their relationships on class diagrams

Figure 21.3 Banking System class context diagram



Object-Oriented Software Life Cycle

Analysis Modeling

- Static Modeling
  - Define structural relationships between classes
  - Depict classes and their relationships on class diagrams

Figure 21.4 Conceptual static model for Banking System - entity classes

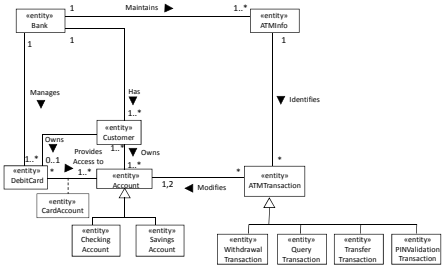
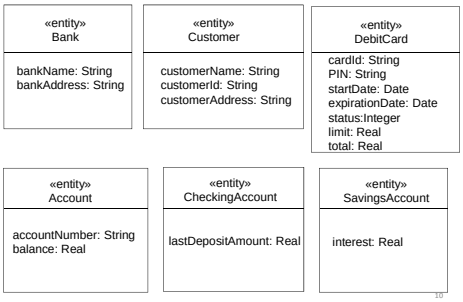


Figure 21.5 Conceptual static model for Banking System

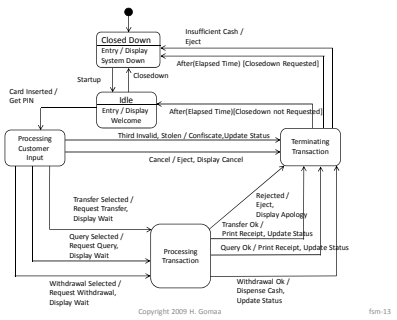


Object-Oriented Software Life Cycle

Analysis Modeling

- **Dynamic Modeling**
  - Define statecharts for state dependent control objects

Figure 21.21 Top level statechart for ATM Control

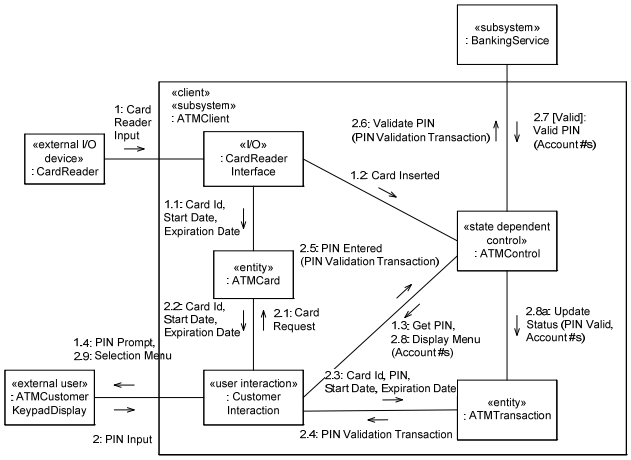


 fsm-13

Object-Oriented Software Life Cycle

Analysis Modeling

- **Dynamic Modeling**
  - Defines how objects participate in use cases using communication diagrams or sequence diagrams



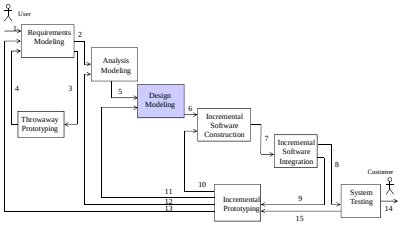
Steps in Using COMET/UML

1 Develop Software Requirements Model

2 Develop Software Analysis Model

3 **Develop Software Design Model**
  - Design Overall Software Architecture (Chapter 12, 13)
  - Design Distributed Component-based Subsystems (Chapter 13,15)
  - Structure Subsystems into Concurrent Tasks (Chapter 18)
  - Design Information Hiding Classes (Chapter 14)
  - Develop Detailed Software Design

Figure 5.1 COMET software life cycle model

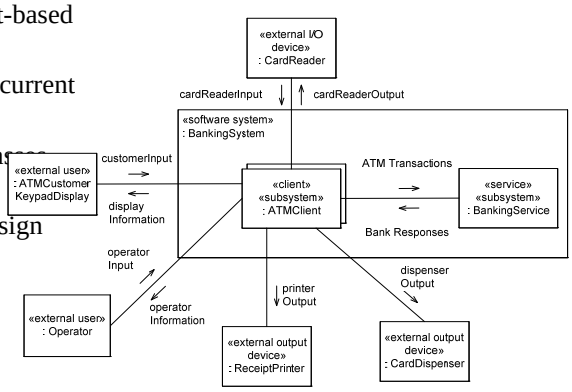


Object-Oriented Software Life Cycle

Design Modeling

- **Design Overall Software Architecture (Chapter 12, 13)**
- Design Distributed Component-based Subsystems (Chapter 13,15)
- Structure Subsystems into Concurrent Tasks (Chapter 18)
- Design Information Hiding Classes (Chapter 14)
- Develop Detailed Software Design

Figure 21.27 Subsystem design – high level communication diagram for Banking System

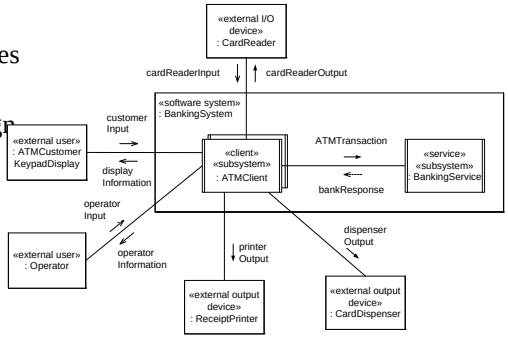


Object-Oriented Software Life Cycle

Design Modeling

- Design Overall Software Architecture (Chapter 12, 13)
- **Design Distributed Component-based Subsystems (Chapter 13,15)**
- Structure Subsystems into Concurrent Tasks (Chapter 18)
- Design Information Hiding Classes (Chapter 14)
- Develop Detailed Software Design

Figure 21.28 Subsystem interfaces – high level communication diagram for Banking System

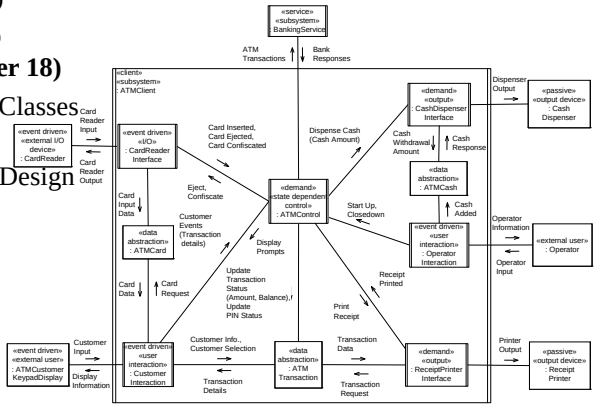


Object-Oriented Software Life Cycle

Design Modeling

- Design Overall Software Architecture (Chapter 12, 13)
- Design Distributed Component-based Subsystems (Chapter 13,15)
- **Structure Subsystems into Concurrent Tasks (Chapter 18)**
- Design Information Hiding Classes (Chapter 14)
- Develop Detailed Software Design

Figure 21.29 Task architecture – initial concurrent communication diagram for ATM Client subsystem

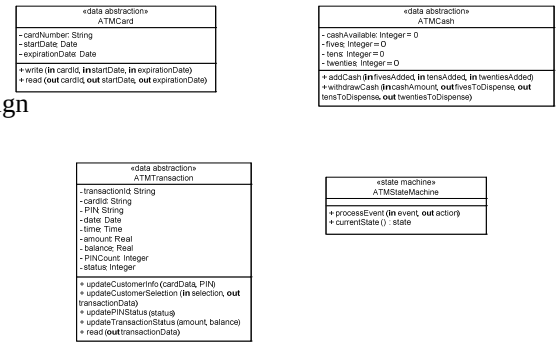


Object-Oriented Software Life Cycle

Design Modeling

- Design Overall Software Architecture (Chapter 12, 13)
- Design Distributed Component-based Subsystems (Chapter 13,15)
- Structure Subsystems into Concurrent Tasks (Chapter 18)
- **Design Information Hiding Classes (Chapter 14)**
- Develop Detailed Software Design

Figure 21.31 ATMClient information hiding classes

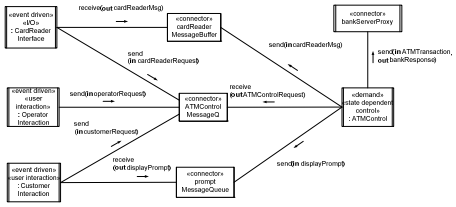


Object-Oriented Software Life Cycle

Design Modeling

- Design Overall Software Architecture (Chapter 12, 13)
- Design Distributed Component-based Subsystems (Chapter 13,15**)**
- Structure Subsystems into Concurrent Tasks (Chapter 18)
- Design Information Hiding Classes (Chapter 14)
- **Develop Detailed Software Design**

Example of cooperating tasks using connectors



Steps in Using COMET/UML

1 Develop Software Requirements Model
- Develop Use Case Model (Chapter 6)

2 Develop Software Analysis Model
- Develop static model of problem domain (Chapter 7)
- Structure system into objects (Chapter 8)
- Develop statecharts for state dependent objects (Chapter 10)
- Develop object interaction diagrams for each use case (Chapter 9, 11)

3 Develop Software Design Model
- Design Overall Software Architecture (Chapter 12, 13)
- Design Distributed Component-based Subsystems (Chapter 13,15)
- Structure Subsystems into Concurrent Tasks (Chapter 18)
- Design Information Hiding Classes (Chapter 14)
- Develop Detailed Software Design