SWE 621:
Software Modeling and Architectural Design

Lecture Notes on Software Design

Lecture 4- Object Structuring

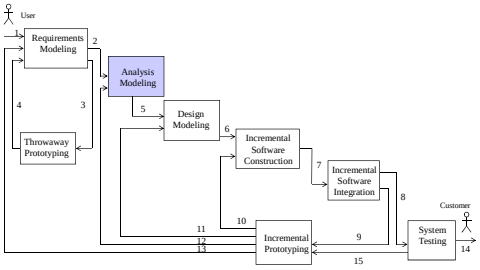
Hassan Gomaa
Dept of Computer Science
George Mason University
Fairfax, VA

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



2

Lecture 4:
Object Structuring

Hassan Gomaa

References: H. Gomaa, Chapter 8 - *Software Modeling and Design*, Cambridge University Press, February 2011

H. Gomaa, Chapter 9 - *Designing Concurrent, Distributed, and Real-Time Applications with UML*, Addison Wesley Object Technology Series, July, 2000

Object Structuring Criteria

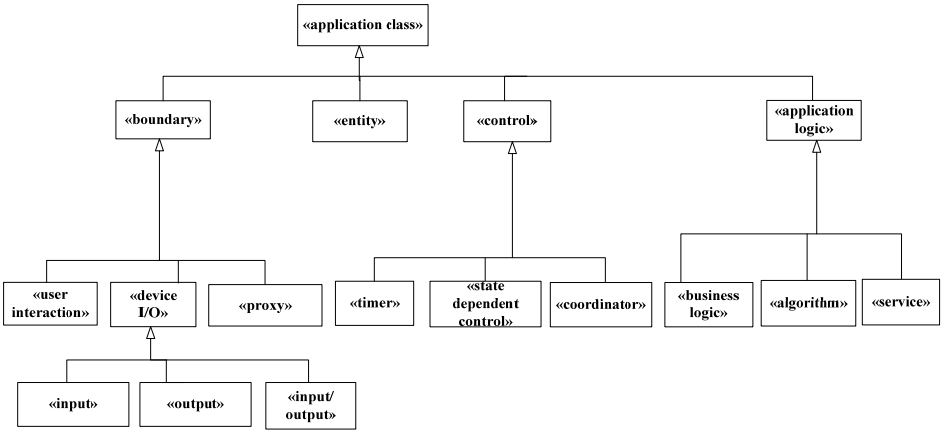
- Determine all software objects in system
  - Use Object Structuring Criteria
  - Guidelines for identifying objects
- Structuring criteria depicted using stereotypes
  - **Stereotype** defines a new building block that is derived from an existing UML modeling element but is tailored to the modeler's problem
  - Depicted using guillemets
    - «entity», «boundary», «control»
- Objects are categorized
  - A **category** is a specifically defined division in a system of classification

Object Structuring Criteria

- Boundary objects
  - User interaction object
  - Device I/O object
  - Proxy object
- Entity objects
  - Long living objects that store information
  - Determined during static modeling
- Control objects
- Application Logic Objects

5

Figure 8.1: Classification of application classes using stereotypes



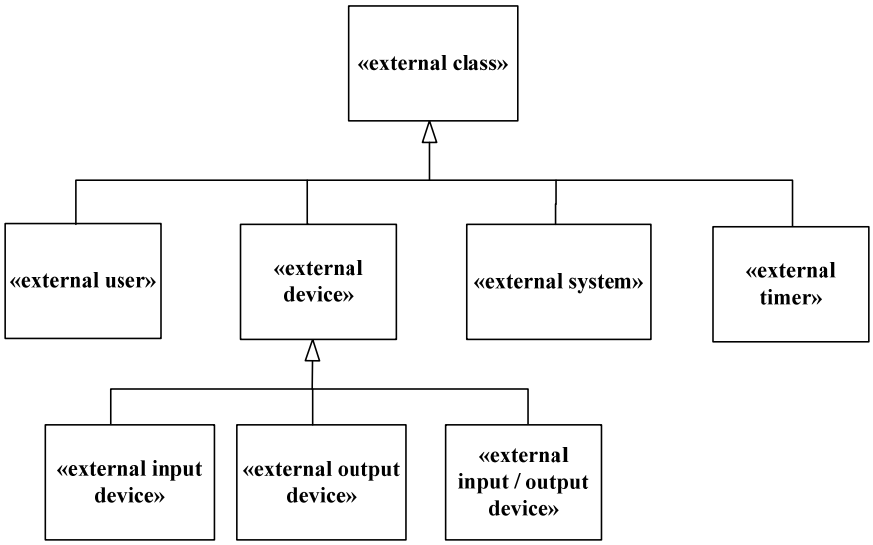
6

Object Structuring Criteria

- Boundary objects
  - Interface to and communicate with external environment
  - Each software boundary object interfaces to an external (real-world) object
    - User interaction object
    - Device I/O object
    - Proxy object
- For each boundary object there is a corresponding external object
  - Also depicted using stereotype

7

Figure 7.22 Classification of external classes using stereotypes



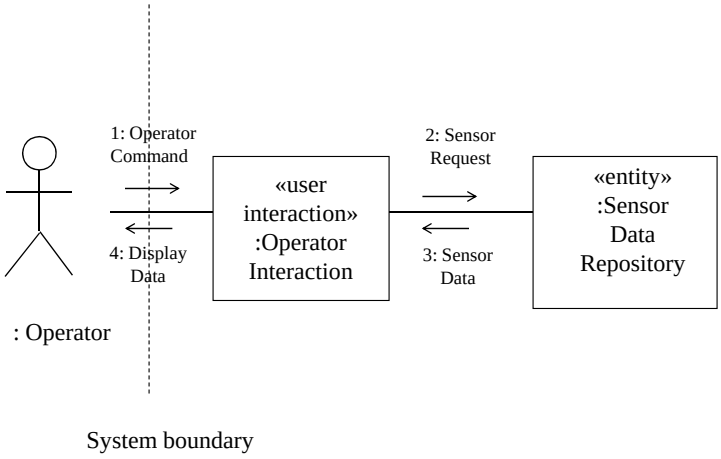
8

User interaction object

- Interfaces to and interacts with a human user
  - Via standard I/O devices
    - keyboard, visual display, mouse
  - Support simple or complex user interfaces
    - Command line interface
    - Graphical user interface (GUI)

9

Figure 8.2 Example of user interaction object



Note: The dashed line for the system boundary is for illustrative purposes only and does not conform to the UML notation

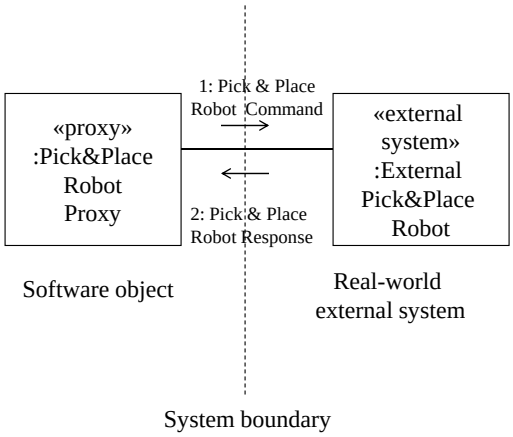
10

Object Structuring Criteria

- Proxy object
  - Interfaces to an external system
  - Hides details of **how** to communicate with external system
    - E.g., Robot Proxy
    - Interfaces to external (real-world) robot

11

Figure 8.3 Example of proxy object



Note: The dashed line for the system boundary is for illustrative purposes only and does not conform to the UML notation

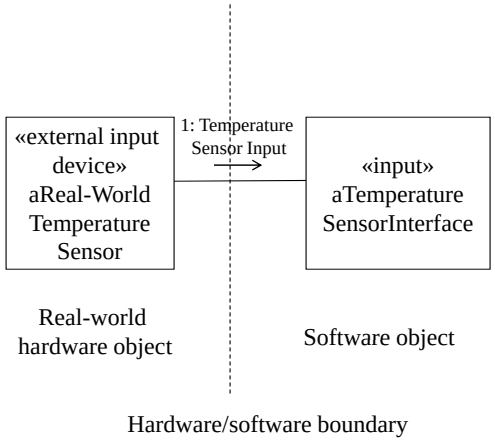
12

Object Structuring Criteria

- Device I/O boundary object
  - Interfaces to I/O device
- Input object
  - E.g., Sensor Interface
- Output object
  - E.g., Actuator Interface
- I/O (Input/Output) object
  - E.g., ATM Card Reader Interface

13

Figure 8.4 Example of input object



Note: The dashed line for the hardware/software boundary is for illustrative purposes only and does not conform to the UML notation

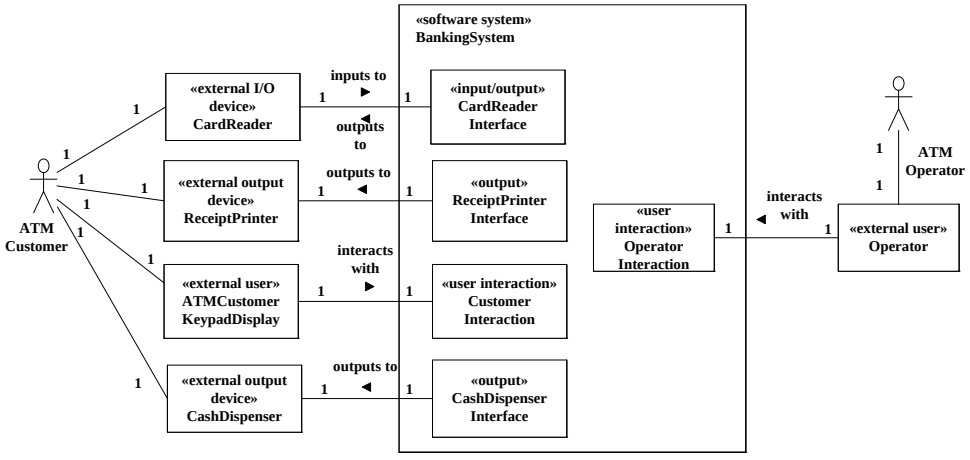
14

Depicting External Classes and Boundary Classes

- Start from system context class diagram
  - Shows external classes
  - System (aggregate class)
- Each **external class** must interface to
  - software **boundary class**
- UML
  - System shown as aggregate class
  - External classes are outside the system class
  - Boundary classes are inside the system class

15

Figure 8.7 Banking System external classes and software boundary classes

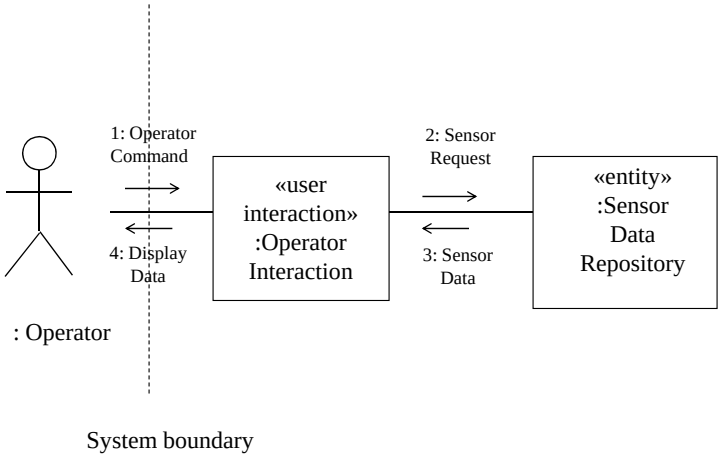


16

Object Structuring Criteria

- Entity objects
  - Long lasting objects that store information
    - Same object typically accessed by many use cases
    - Information persists over access by several use cases
      - E.g., Account, Customer
  - Entity classes and relationships shown on static model
  - Entity classes often mapped to relational database during design

Figure 8.2 Example of entity object



Note: The dashed line for the system boundary is for illustrative purposes only and does not conform to the UML notation

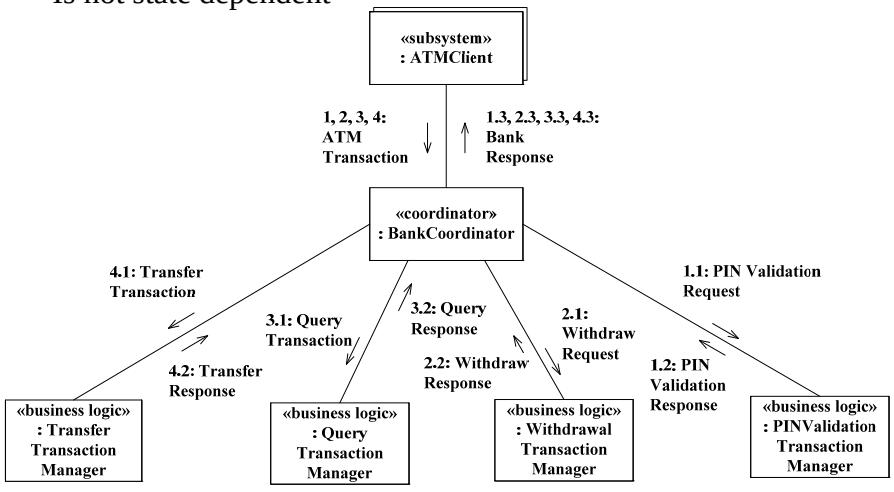
Object Structuring Criteria

- Control objects
  - Provides overall coordination for execution of a group of objects
  - Makes overall decisions
  - Decides when, and in what order, other objects participate in interaction sequence
    - Entity objects
    - Boundary objects
- Control objects
  - Coordinator object
  - State dependent control object
  - Timer object

19

Figure 8.10 Coordinator object

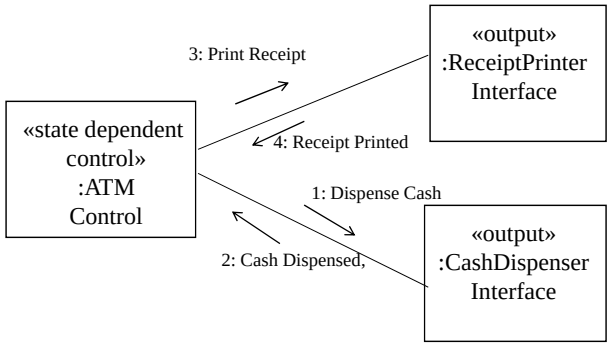
Coordinator object
- Provides sequencing for group of objects
- Is not state dependent



20

Figure 8.11 State dependent control object

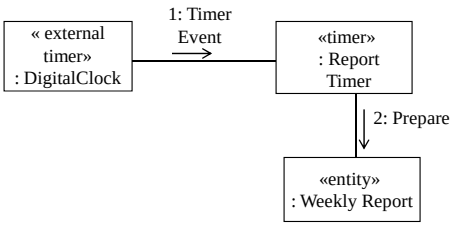
State dependent control object
- Defined by finite state machine or state transition table
- Controls other objects



21

Figure 8.12 Timer object

Timer object
- Activated periodically



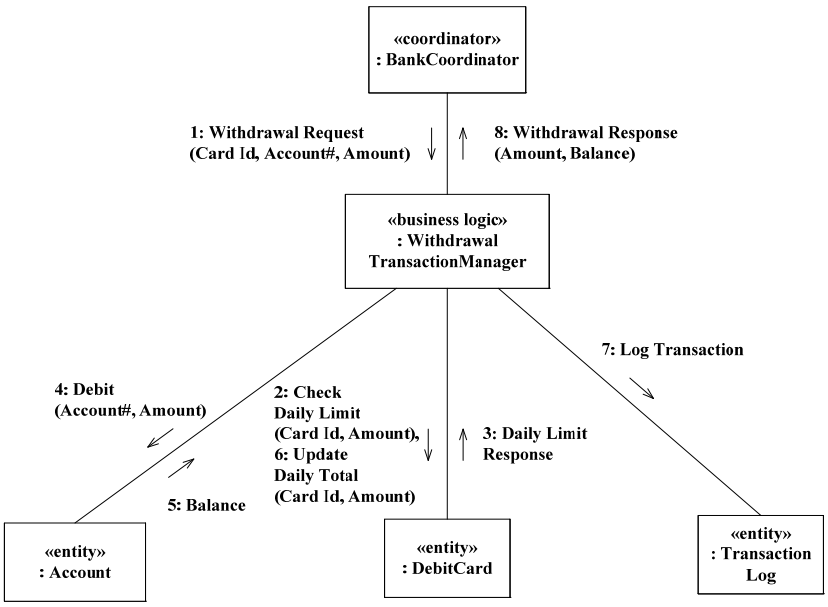
22

Object Structuring Criteria

- Application Logic Objects
  - Business Logic Object
    - Defines business specific application logic (rules) for processing a client request
    - Usually accesses more than one entity object
  - Algorithm Object
  - Service object

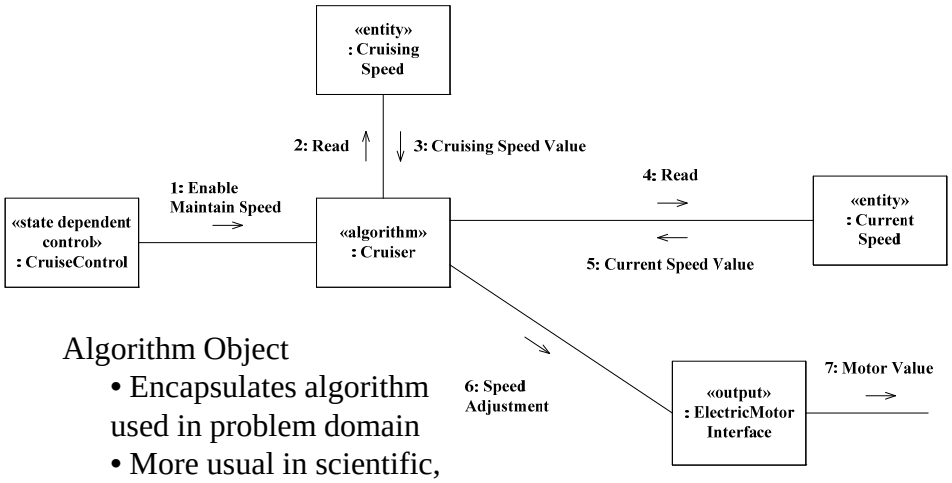
23

Figure 8.13b Business logic object



24

Figure 8.14 Example of algorithm object

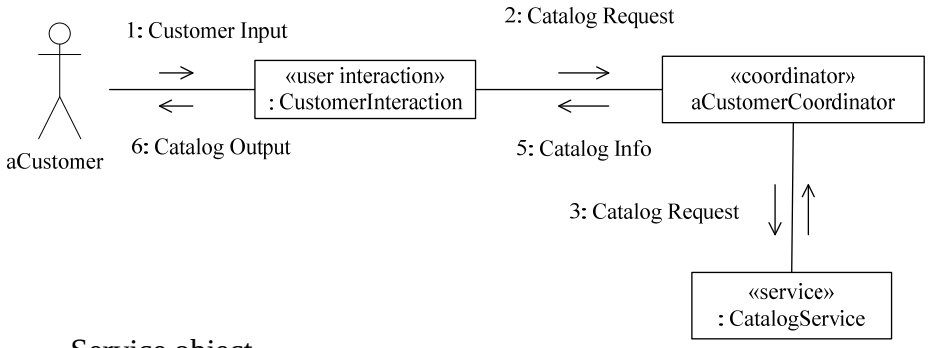


Algorithm Object

- Encapsulates algorithm used in problem domain
- More usual in scientific, engineering, real-time domains

25

Figure 8.15 Example of Service object



Service object

- Provides a service to other objects
- Responds to requests from client objects

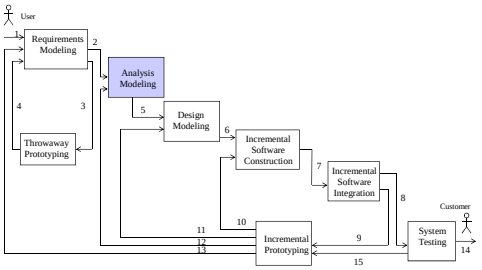
26

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



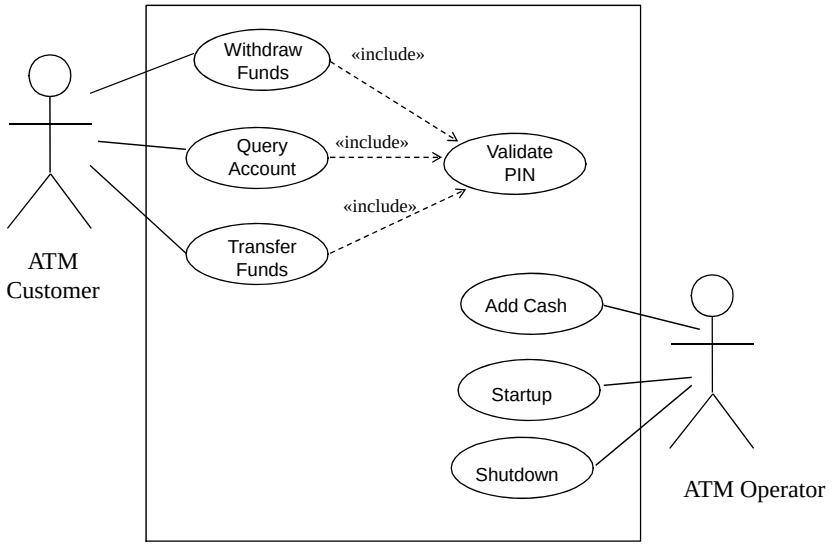
27

Case Study: Banking System

- Multiple Automated Teller Machines (ATM)
  - Customer inserts ATM Card
  - Enters Personal Identification Number (PIN)
  - ATM Transactions
    - PIN Validation
    - Withdraw Funds from Checking or Savings Account
    - Query Account
    - Transfer funds between accounts
- Banking System maintains information about
  - Customers
  - Debit cards
  - Checking and Savings Accounts

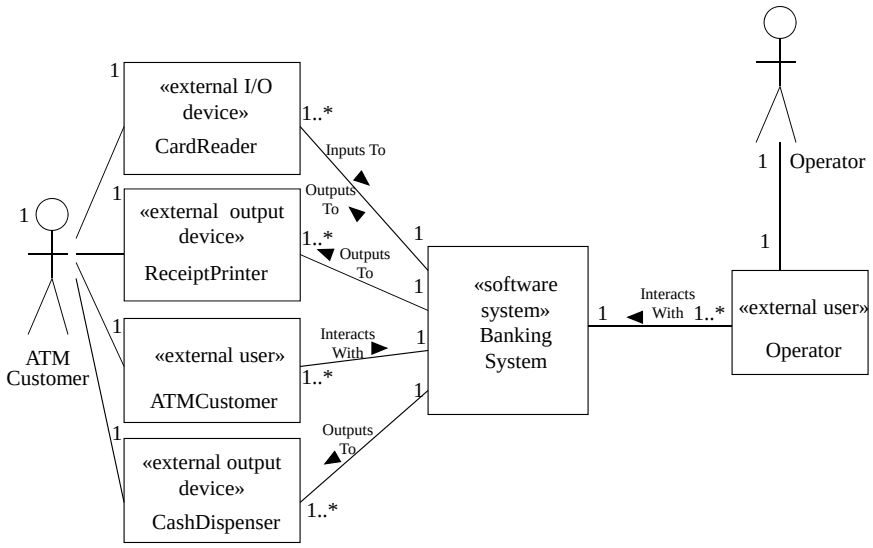
28

Figure 21.1 Banking System use case model



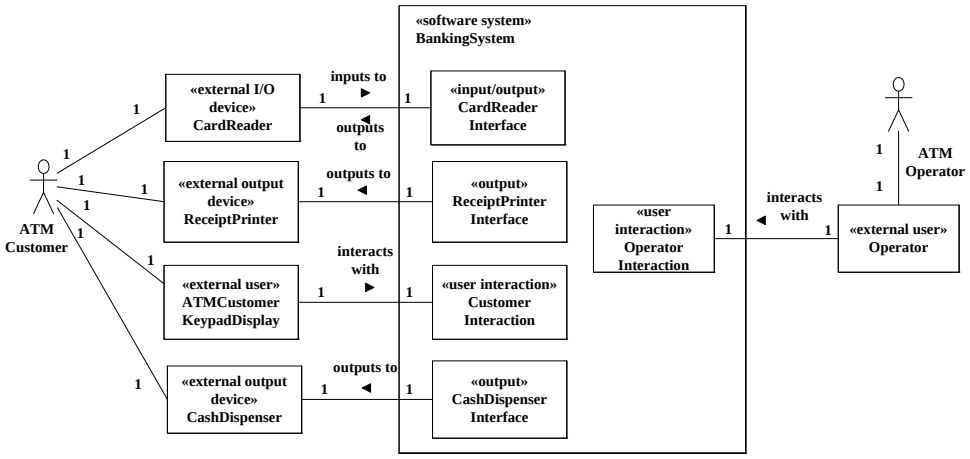
29

Figure 21.3 Banking System class context diagram



30

Figure 21.9 Banking System external classes and software boundary classes



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