Object and Class Structuring

Chapter 9

Part of Analysis Modeling

Designing Concurrent, Distributed, and Real-Time Applications with UML
Hassan Gomaa (2001)

Object and Class Structuring

- First attempt at determining the software objects in the system.
  - Emphasis on objects modeling real-world and entity objects
- Determines and Categorizes the software interfaces to the objects developed in the Static Modeling phase.
- The beginnings of the Dynamic Model

Object Structuring Criteria

- No uniquely correct decompositions of a system.
  - Analytical Judgment
  - Problem Characteristics
  - Problem Domain
- Provide guidance for structural decisions.

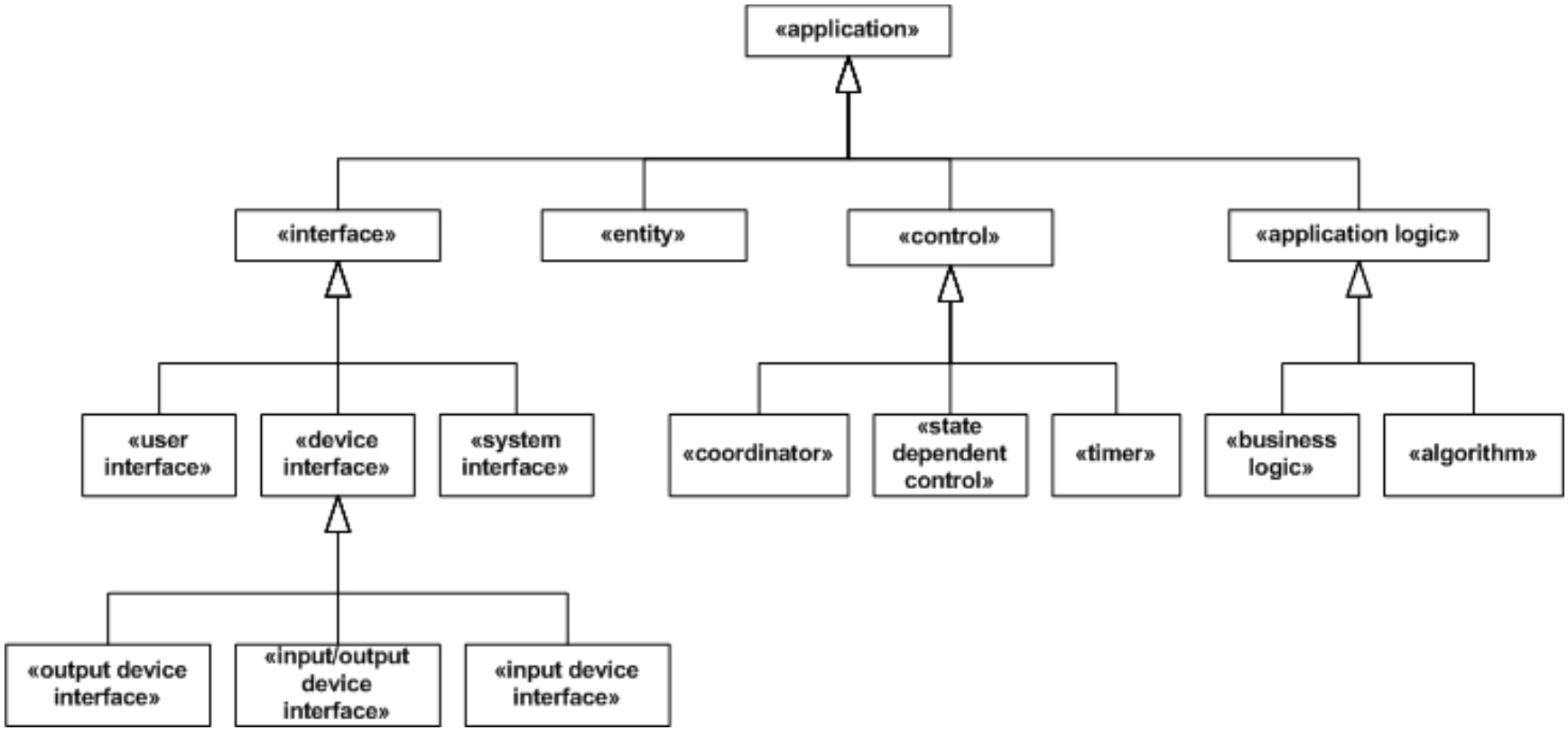
Categorizing Application Classes

- Recall that *objects* are instances of *classes.*
- Interface Objects
  - Represent connections to the external environment of the system.
  - Device Interface Objects connect to a hardware I/O device.
  - User Interface Objects provide for human user interaction.
  - System Interface Objects represent external systems or subsystems.
- Entity Objects
  - Long living information stores.

Categorizing Application Classes (cont)

- Control Objects
  - Provide coordination for other objects
    - State-dependant Control Objects
    - Timer Control Objects
    - Coordinator Objects
- Application Logic Objects
  - Contain application details
  - Business Logic Objects represent the general logic of the system.
  - Algorithm Objects encapsulate any special algorithms that may be used by the system.

Categorizing Application Classes (cont)

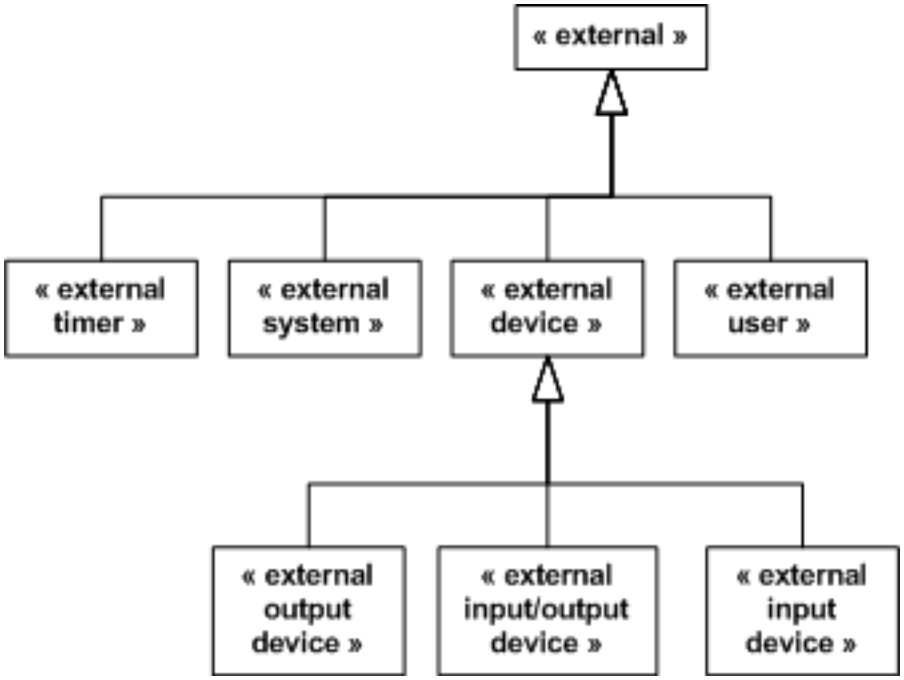


External and Interface Classes

- External classes represent objects outside the system.
- Interface classes represent the connection between external classes and the system.
- Interface classes are part of the system; External classes are not.

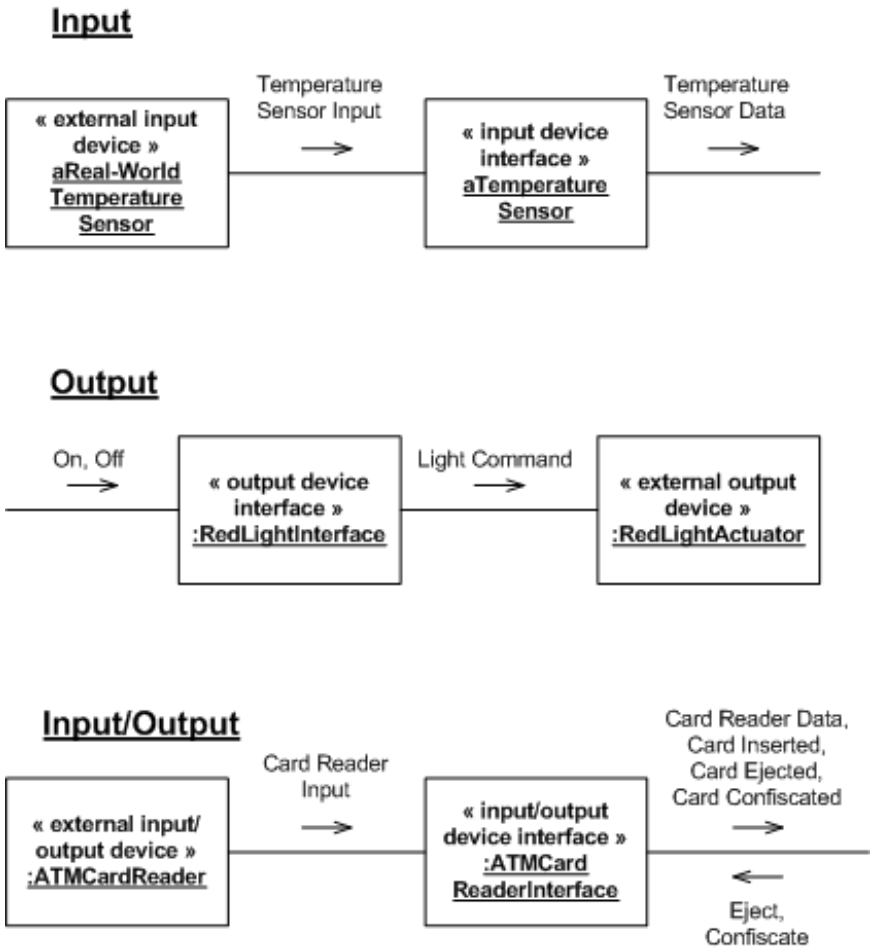
Categorizing External Classes

- Timers
- Devices
  - Input
  - Output
  - Input/Output
- Systems
- Users



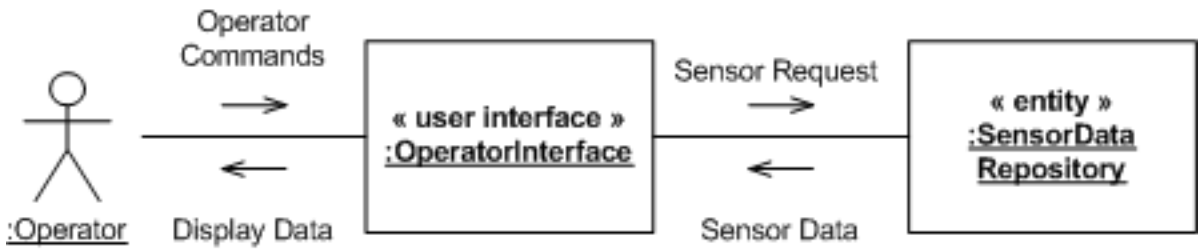
Device Interface Objects

- Represent external sensors, actuators, etc.



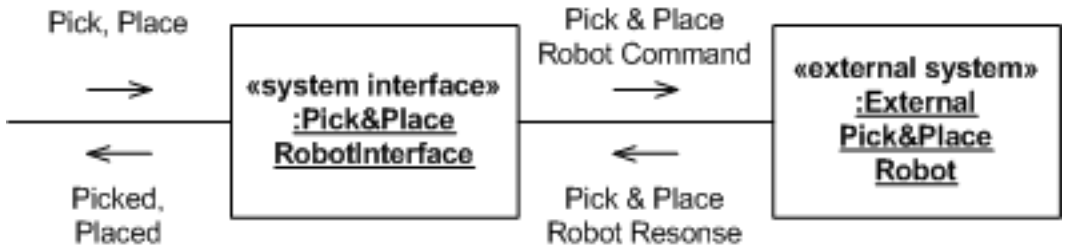
User Interface Objects

- Generally assume the presence of standard I/O objects controlled by the operating system.
- May be complex GUI's or simple CLI's
- May be a composite of many smaller smaller user interfaces.

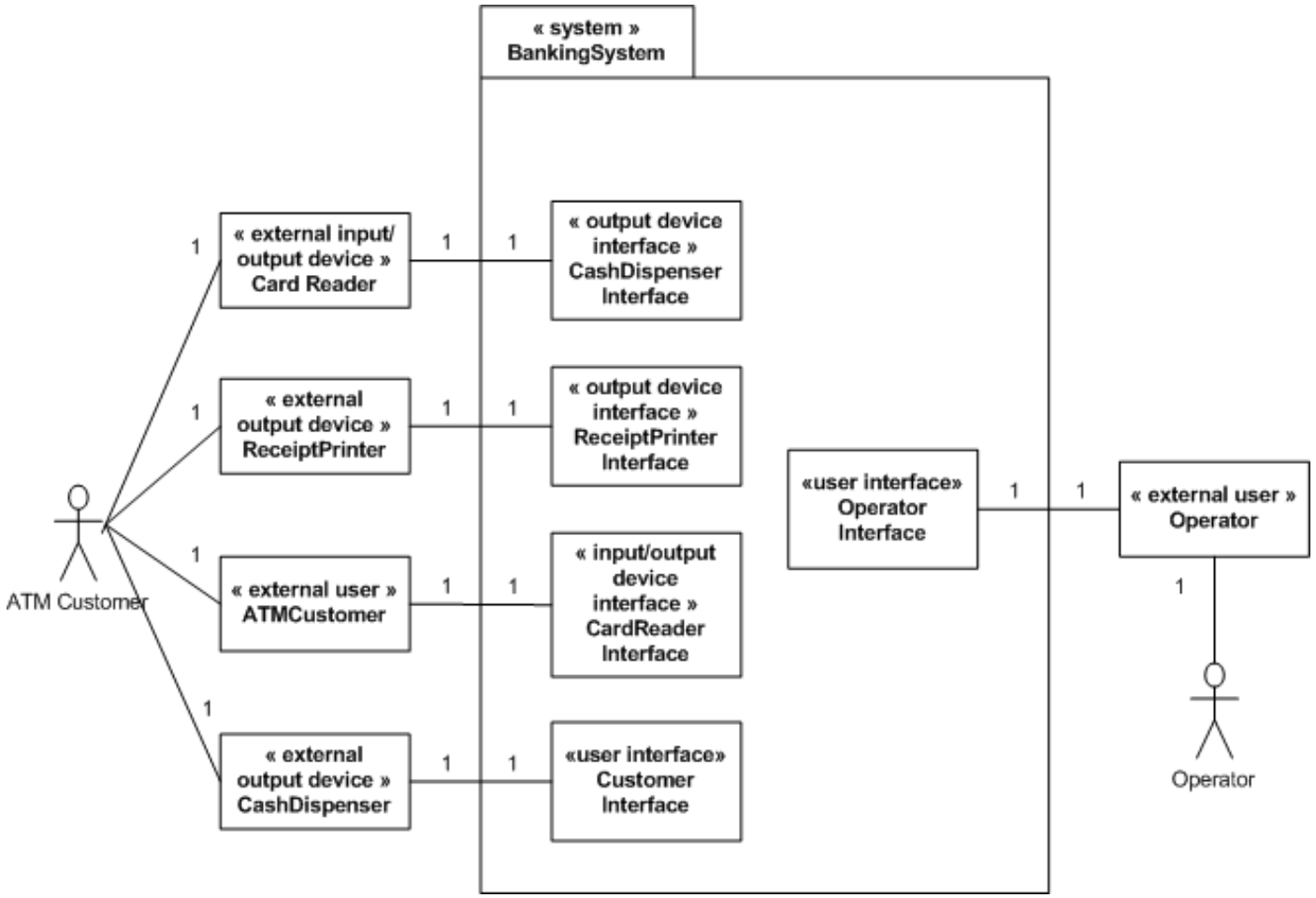


System Interface Objects

- Represent connections to other systems:
  - External Systems
  - Subsystems

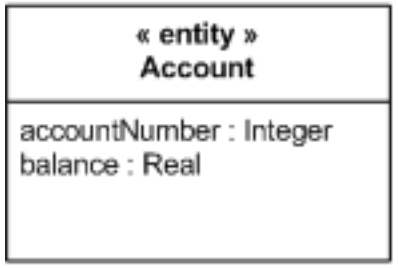


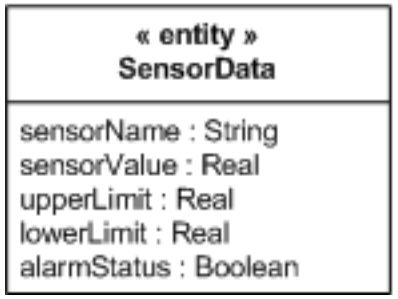
Interface and External Objects

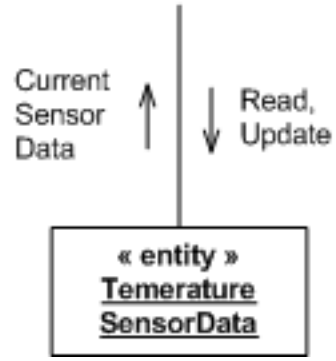


Entity Objects

- Recall that Entity Objects are long living.
  - Used by many use cases.
  - Store information that persists across use cases.
- Entity objects are instances of the entity classes developed in the Static Modeling phase.
- Encapsulate data to limit access.



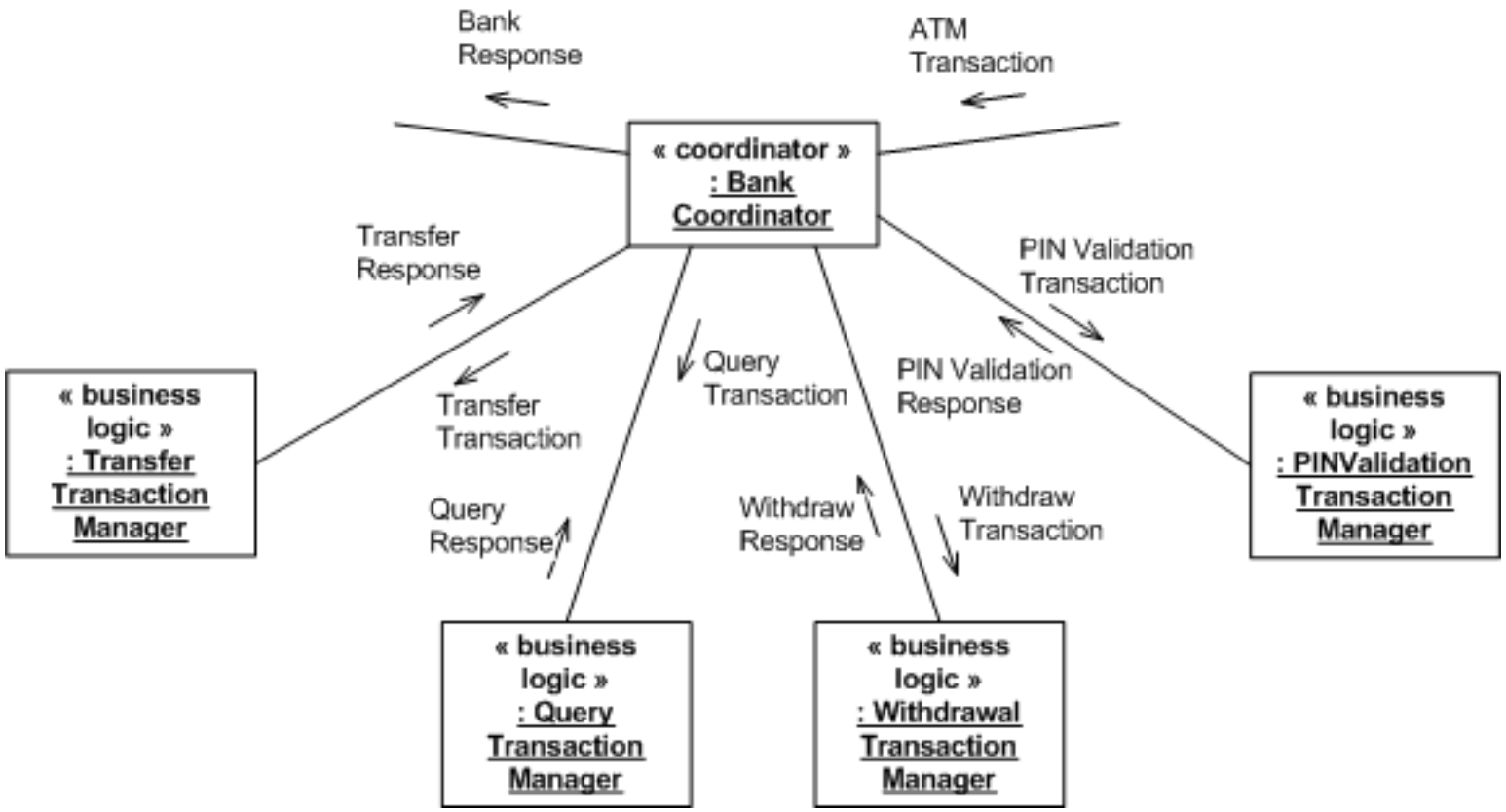




Coordinator Control Objects

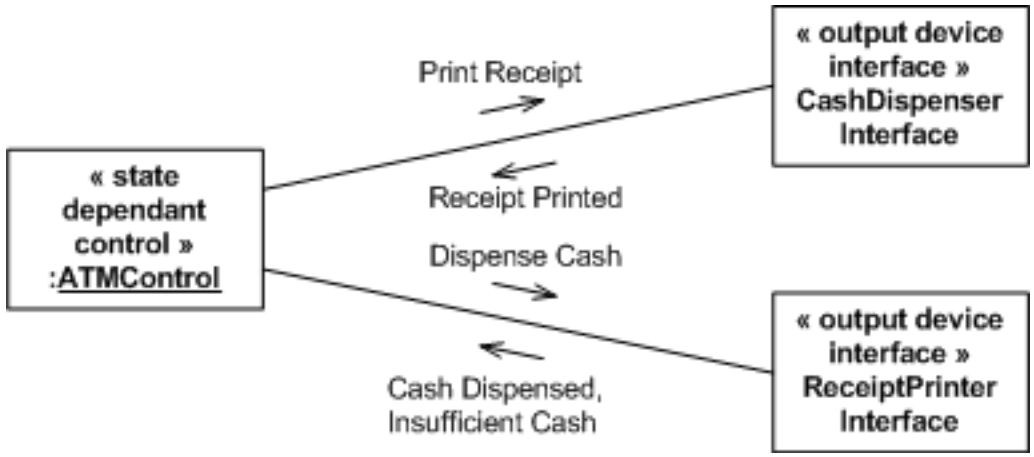
- Make decisions regarding overall sequencing for a collection of objects.
  - Coordinate the actions required for a use case.
  - Decides purely on the input given.
    - i.e. not a state machine.

Coordinator Control Objects (example)



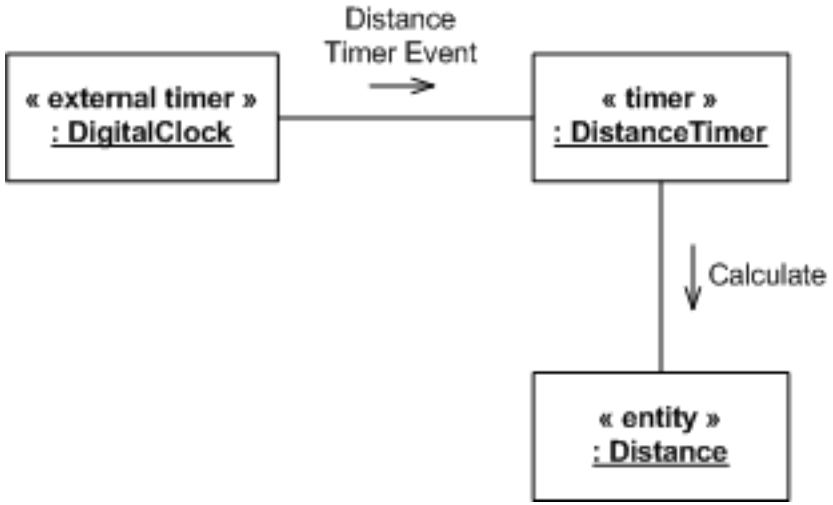
State-Dependant Control Objects

- Make decisions based on both inputs at current state.
  - The behavior changes based on the state of the control object.



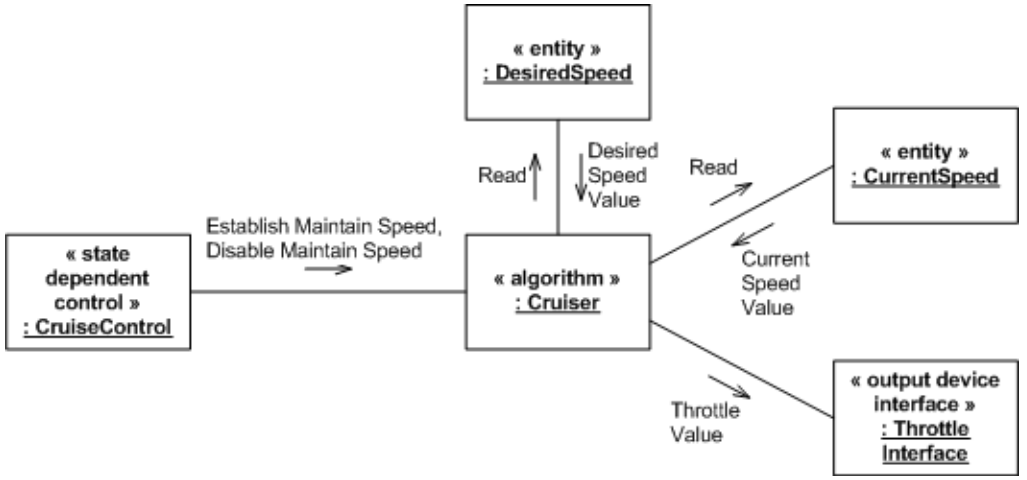
Timer Control Objects

- Timer Control Objects are activated by external timers.
- Timer Control Objects either perform an action themselves, or activate another object to perform the action.



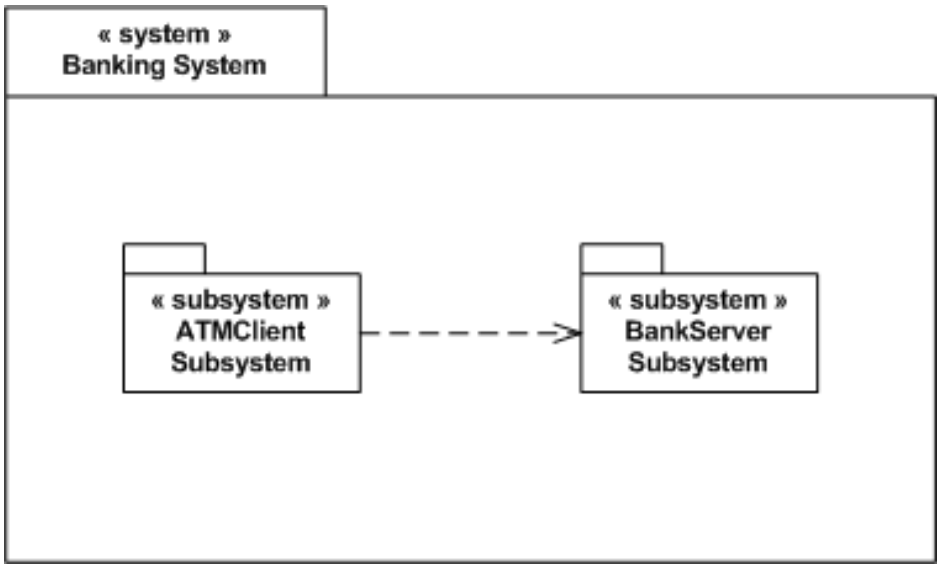
Application Logic Objects

- Business Logic Objects
  - Not always necessary for very simple business logic.
- Algorithm Objects
  - Encapsulates special algorithms
  - Allows the algorithm to change with minimal effort if needed.

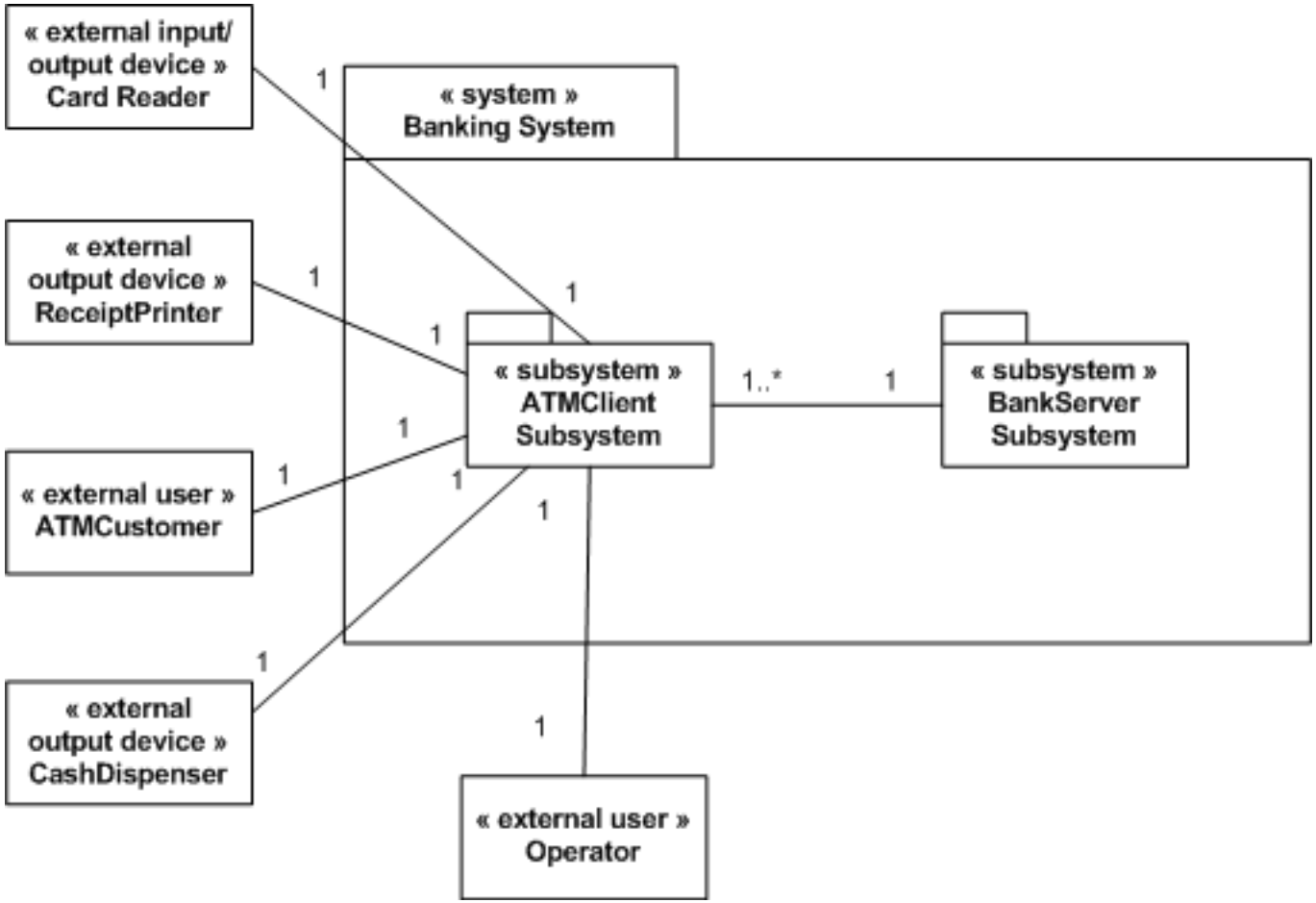


Subsystems

- Objects that share functional dependencies are best placed in a subsystem.
- The Use Cases can be a good place to start if the subsystem divisions are not obvious.
- In COMET, packages are used to show subsystems.



Subsystems Example



Summary

- Application Classes can be categorized into 4 broad categories:
  - Interfaces
  - Entities
  - Controls
  - Application Logic
- The relationship between application classes can be represented using collaboration diagrams

Summary (cont)

- Subsystems can be used to group objects for increased abstraction.
  - Objects can be separated into packages by Use Case if no other divisions are readily available.