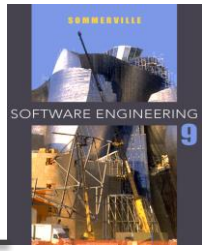
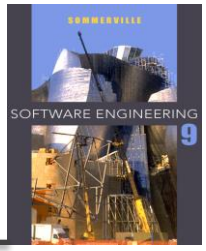


Chapter 7 – Design and Implementation

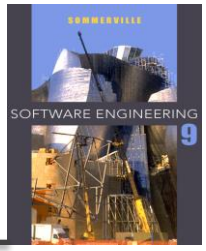
Lecture 1

Topics covered



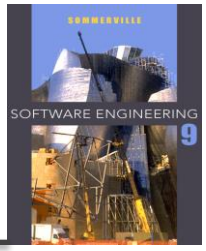
- ✧ Object-oriented design using the UML
- ✧ Implementation issues
- ✧ Open source development

Design and implementation



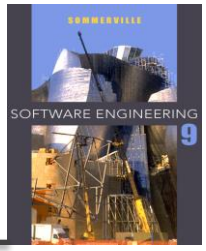
- Software design and implementation is the stage in the software engineering process at which an executable software system is developed.
- Software design and implementation activities are invariably inter-leaved.
  - Software design is a creative activity in which you identify software components and their relationships, based on a customer's requirements.
  - Implementation is the process of realizing the design as a program.

Build or buy



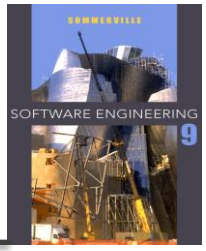
- In a wide range of domains, it is now possible to buy off-the-shelf systems (COTS) that can be adapted and tailored to the users' requirements.
  - For example, if you want to implement a medical records system, you can buy a package that is already used in hospitals. It can be cheaper and faster to use this approach rather than developing a system in a conventional programming language.
- When you develop an application in this way, the design process becomes concerned with how to use the configuration features of that system to deliver the system requirements.

An object-oriented design process



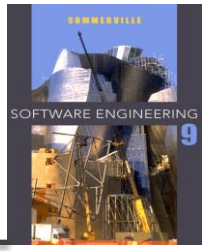
- Structured object-oriented design processes involve developing a number of different system models.
- They require a lot of effort for development and maintenance of these models and, for small systems, this may not be cost-effective.
- However, for large systems developed by different groups design models are an important communication mechanism.

System modeling



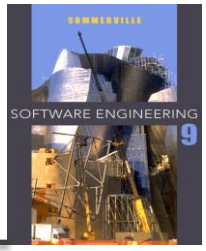
- System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system.
- System modeling has now come to mean representing a system using some kind of graphical notation, which is now almost always based on notations in the Unified Modeling Language (UML).
- System modelling helps the analyst to understand the functionality of the system and models are used to communicate with customers.

System perspectives



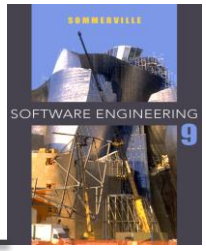
- An external perspective, where you model the context or environment of the system.
- An interaction perspective, where you model the interactions between a system and its environment, or between the components of a system.
- A structural perspective, where you model the organization of a system structure of data flow.
- A behavioral perspective, where you model the dynamic behavior of the system and how it responds to events.

Process stages



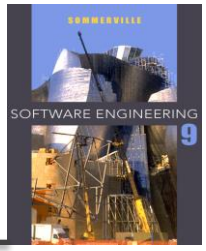
- There are a variety of different object-oriented design processes that depend on the organization using the process.
- Common activities in these processes include:
  - Define the context and modes of use of the system;
  - Design the system architecture;
  - Identify the principal system objects;
  - Develop design models;
  - Specify object interfaces.
- Process illustrated here using a design for a wilderness weather station.

System context and interactions



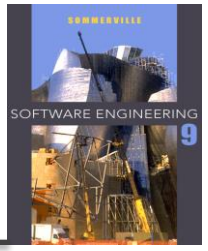
- Understanding the relationships between the software that is being designed and its external environment is essential for deciding how to provide the required system functionality and how to structure the system to communicate with its environment.
- Understanding of the context also lets you establish the boundaries of the system. Setting the system boundaries helps you decide what features are implemented in the system being designed and what features are in other associated systems.

Context and interaction models



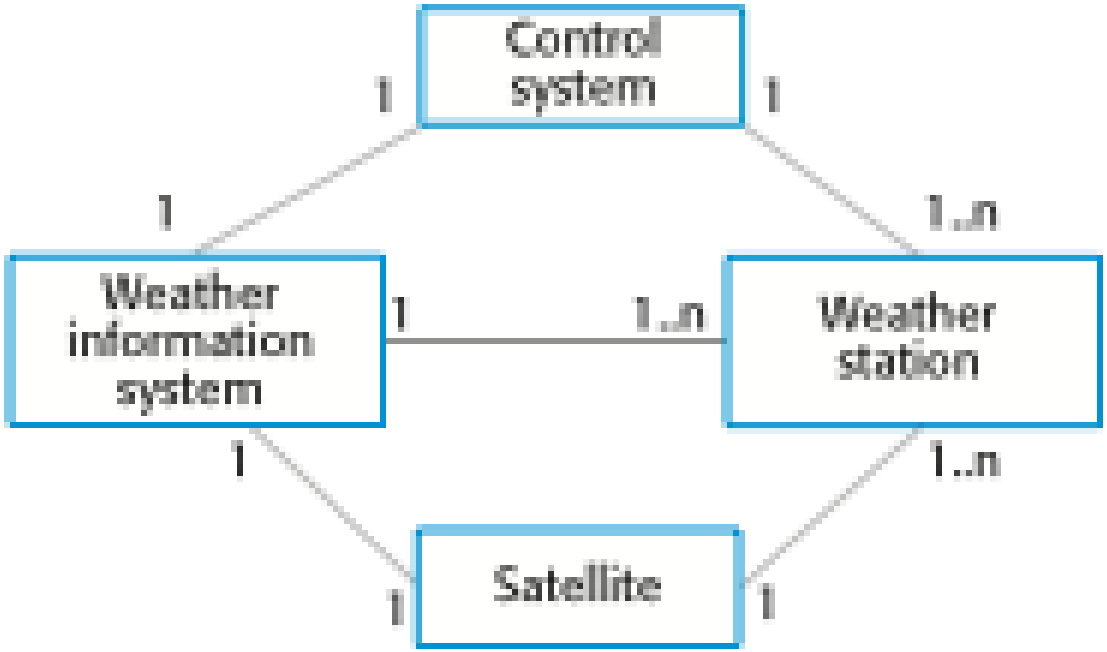
- A system context model is a structural model that demonstrates the other systems in the environment of the system being developed.
- An interaction model is a dynamic model that shows how the system interacts with its environment as it is used.

System boundaries

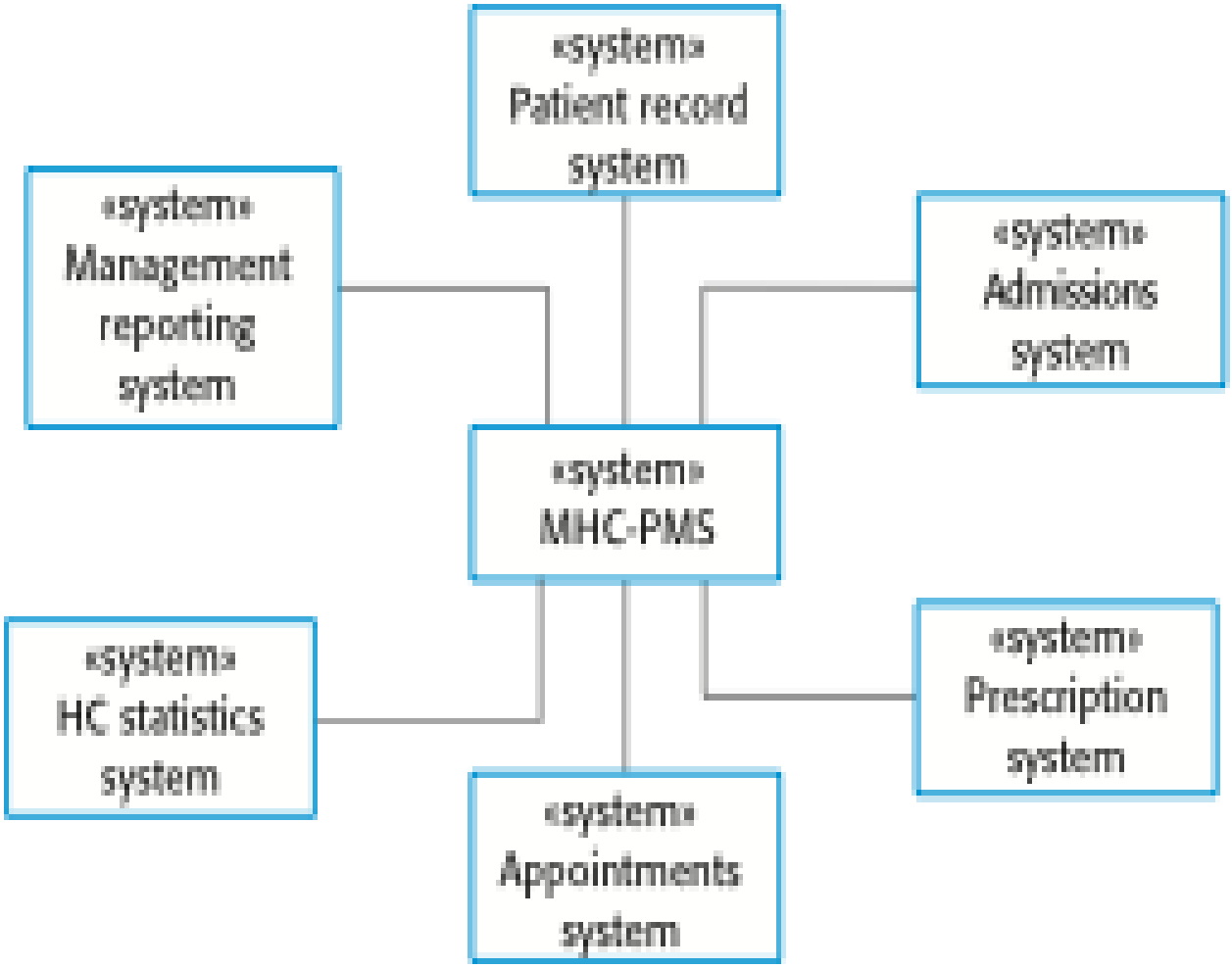


- System boundaries are established to define what is inside and what is outside the system.
  - They show other systems that are used or depend on the system being developed.
- The position of the system boundary has a profound effect on the system requirements.
- Defining a system boundary is a political judgment
  - There may be pressures to develop system boundaries that increase / decrease the influence or workload of different parts of an organization.

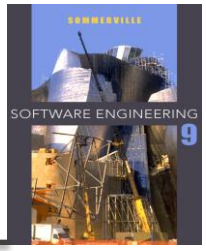
System context for the weather station



The context of the MHC-PMS

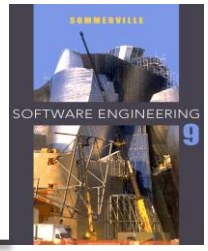


Use case modeling



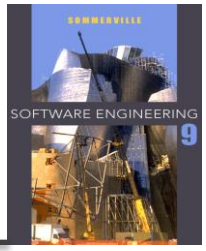
- Use cases were developed originally to support requirements elicitation and now incorporated into the UML.
- Each use case represents a discrete task that involves external interaction with a system.
- Actors in a use case may be people or other systems.
- Represented diagrammatically to provide an overview of the use case and in a more detailed textual form.

Transfer-data use case



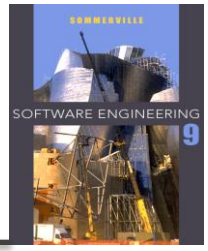
✧ A use case in the MHC-PMS

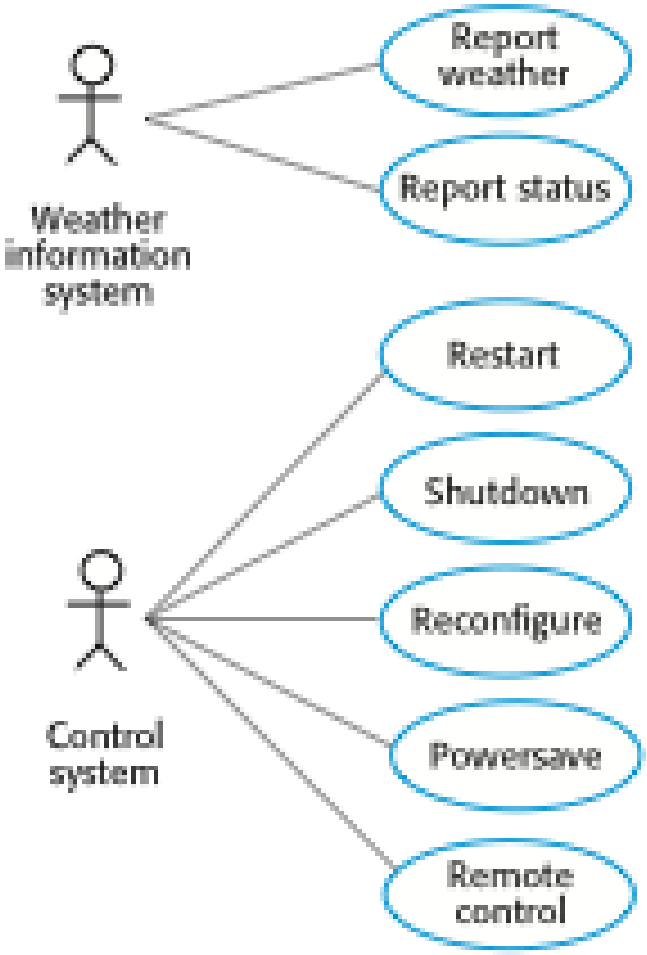
Tabular description of the ‘Transfer data’ use-case



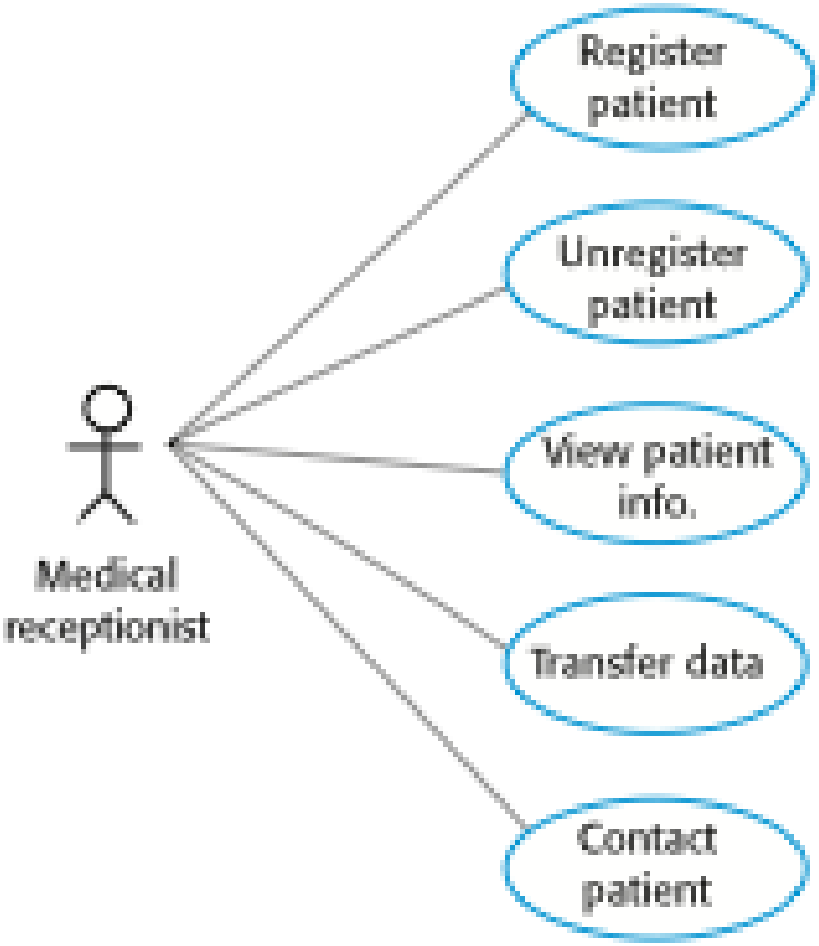
MHC-PMS: Transfer data	
Actors	Medical receptionist, patient records system (PRS)
Description	A receptionist may transfer data from the MHC-PMS to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient’s diagnosis and treatment.
Data	Patient’s personal information, treatment summary
Stimulus	User command issued by medical receptionist
Response	Confirmation that PRS has been updated
Comments	The receptionist must have appropriate security permissions to access the patient information and the PRS.

Weather station use cases

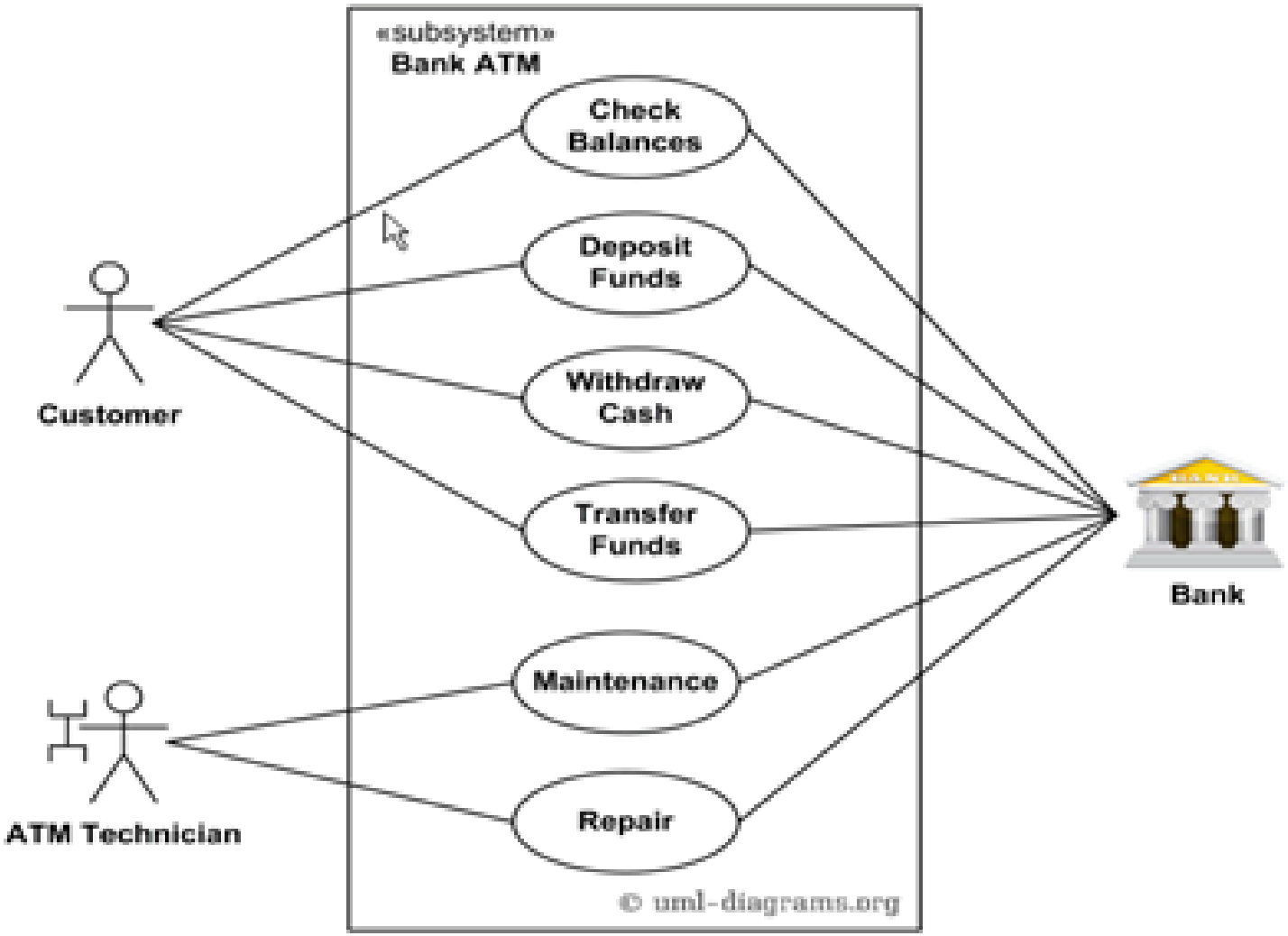


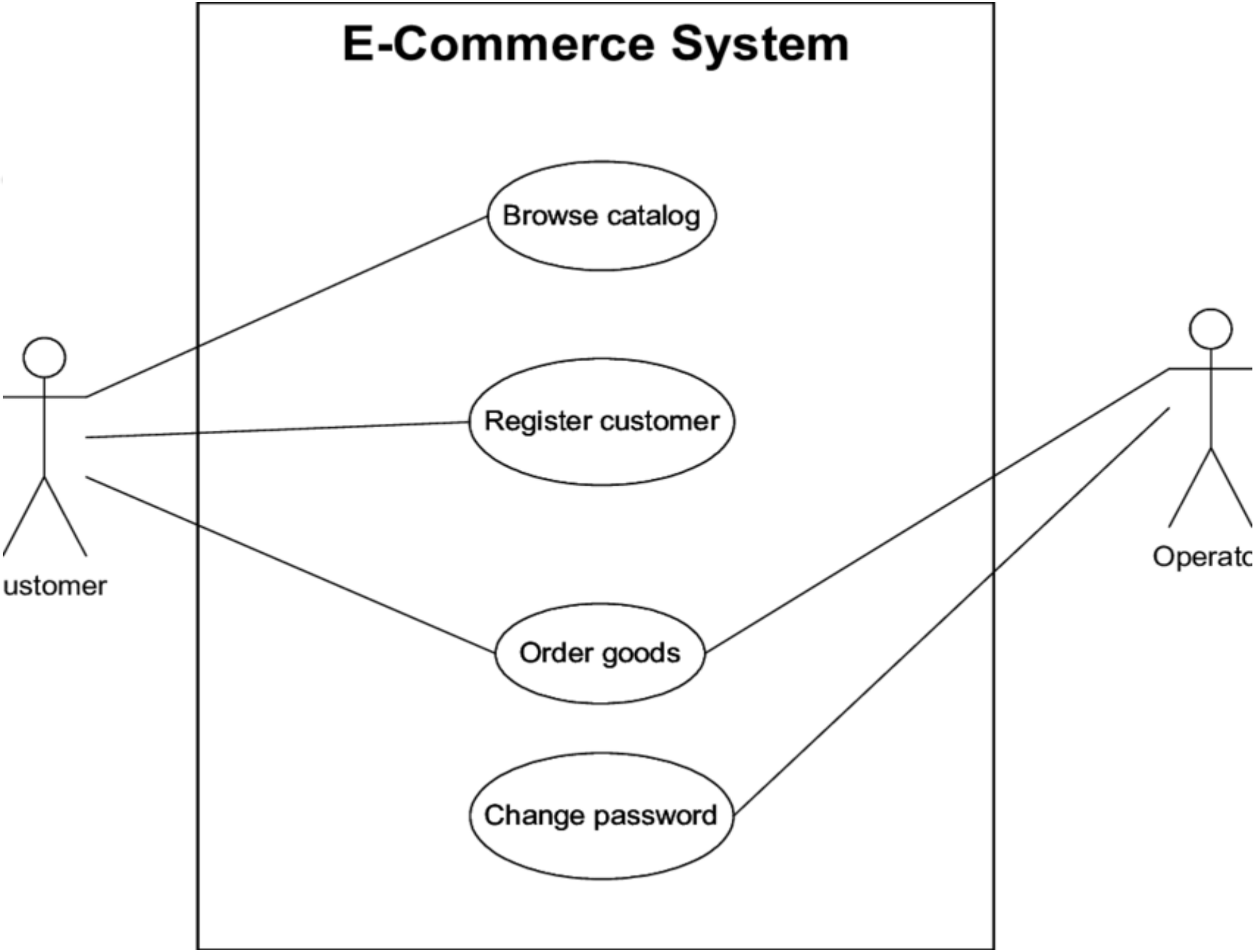


Use cases in the MHC-PMS involving the role 'Medical Receptionist'

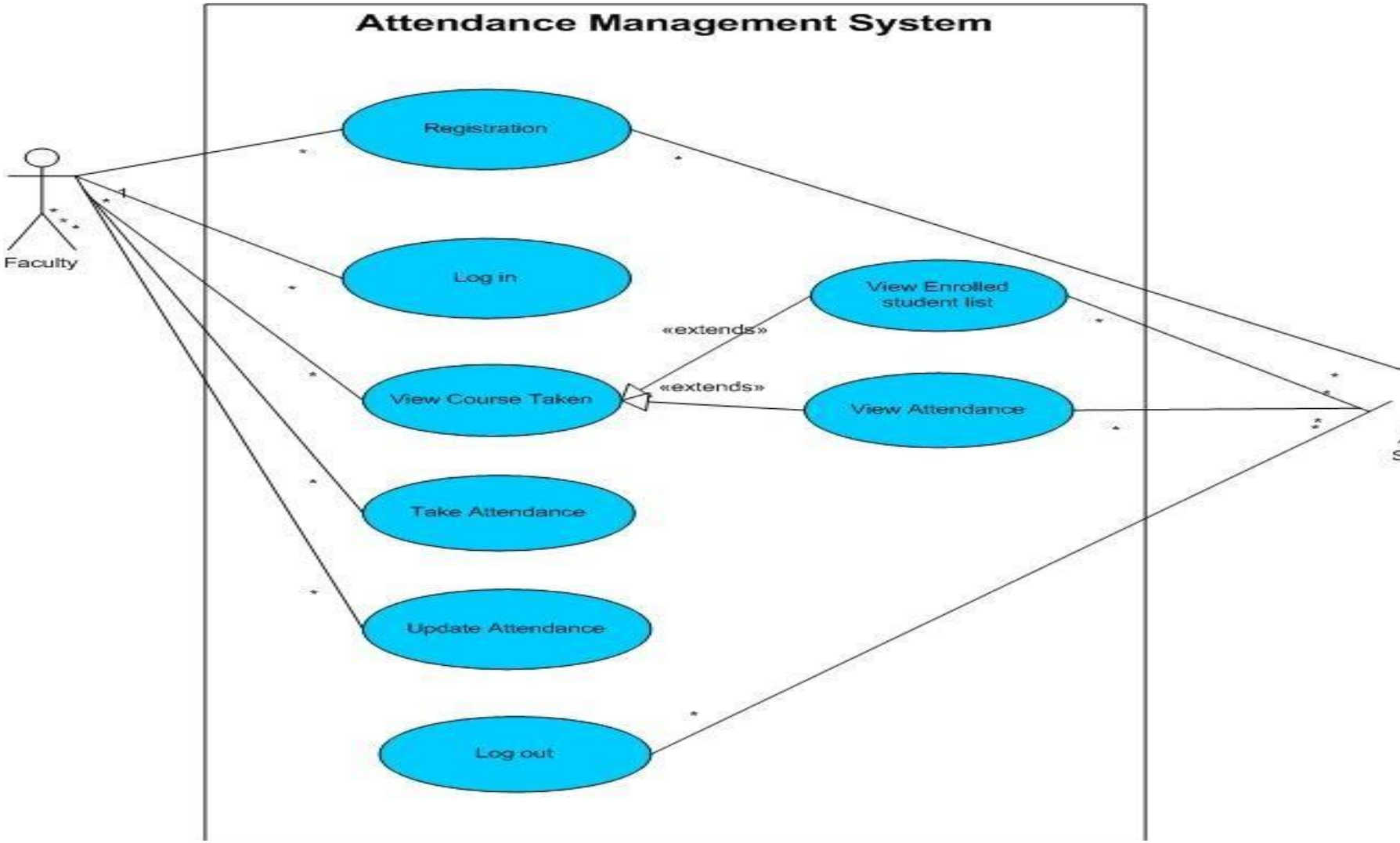


Design a Use-case diagram for Bank ATM system.

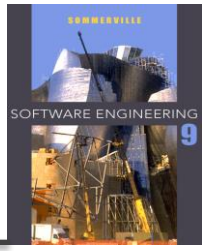



E-Commerce System
Browse catalog
Register customer
Order goods
Change password
ustomer
Operatc

Attendance Management System

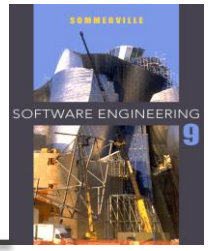


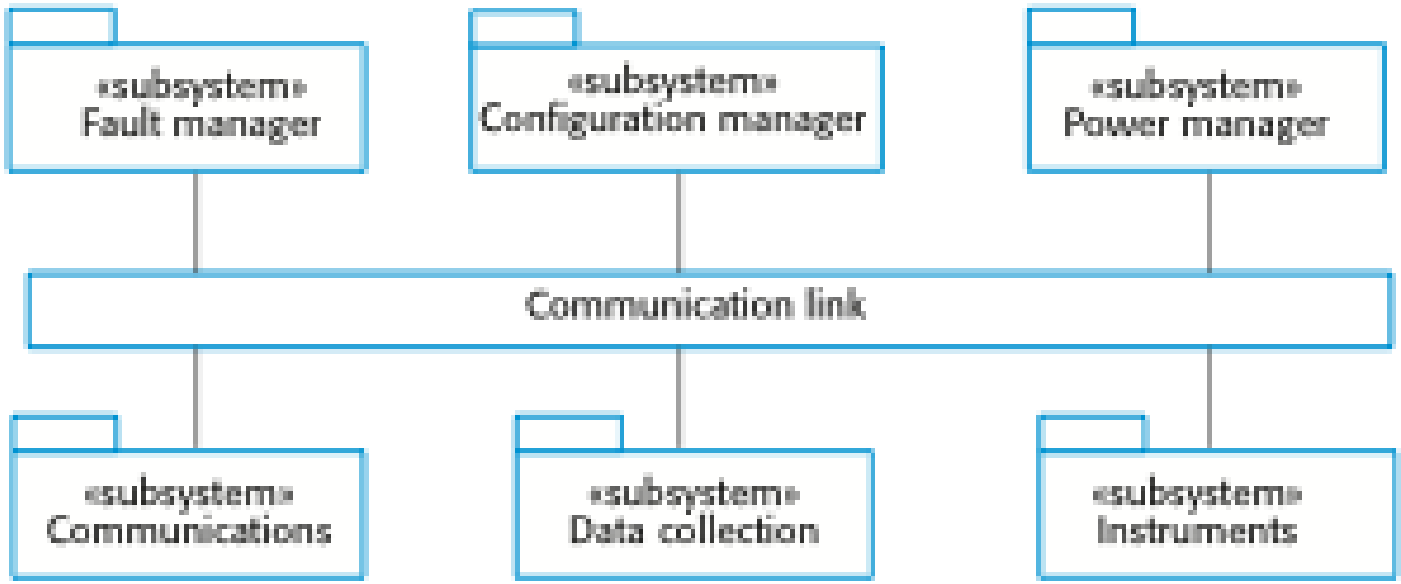
Architectural design



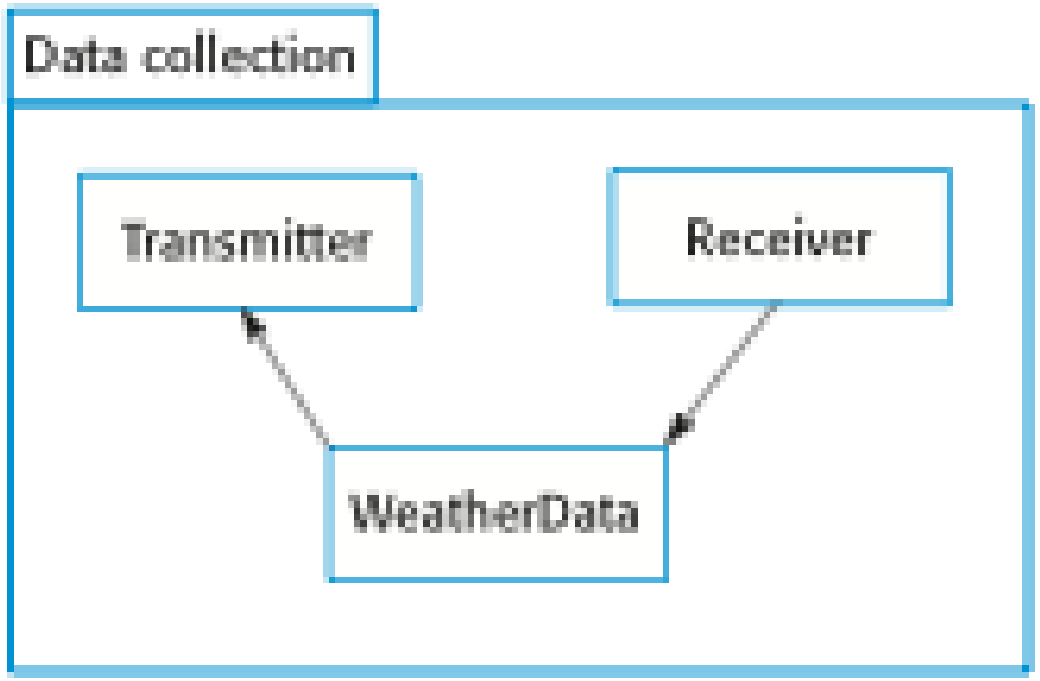
- Once interactions between the system and its environment have been understood, you use this information for designing the system architecture.
- You identify the major components that make up the system and their interactions, and then may organize the components using an architectural pattern such as a layered or client-server model.
- The weather station is composed of independent subsystems that communicate by broadcasting messages on a common infrastructure.

High-level architecture of the weather station

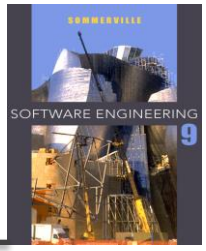




Architecture of data collection system



Object class identification



- Identifying object classes is to often a difficult part of object oriented design.
- There is no 'magic formula' for object identification. It relies on the skill, experience and domain knowledge of system designers.
- Object identification is an iterative process. You are unlikely to get it right first time.

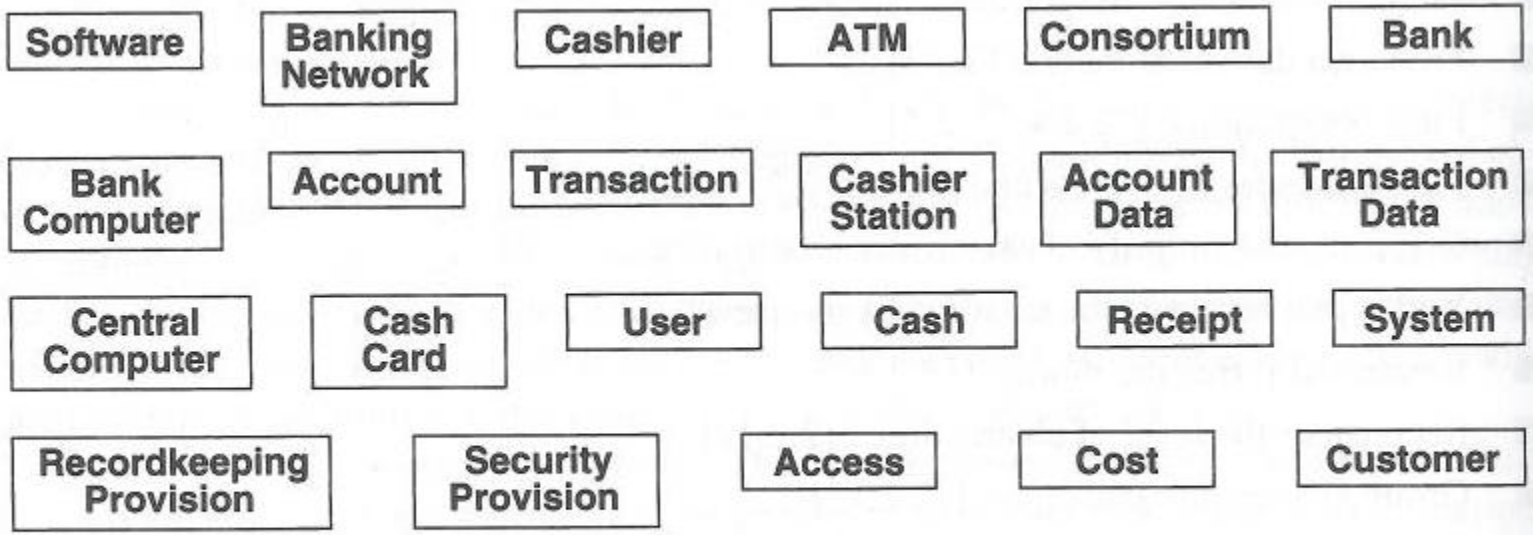


Figure 12.3 ATM classes extracted from problem statement nouns

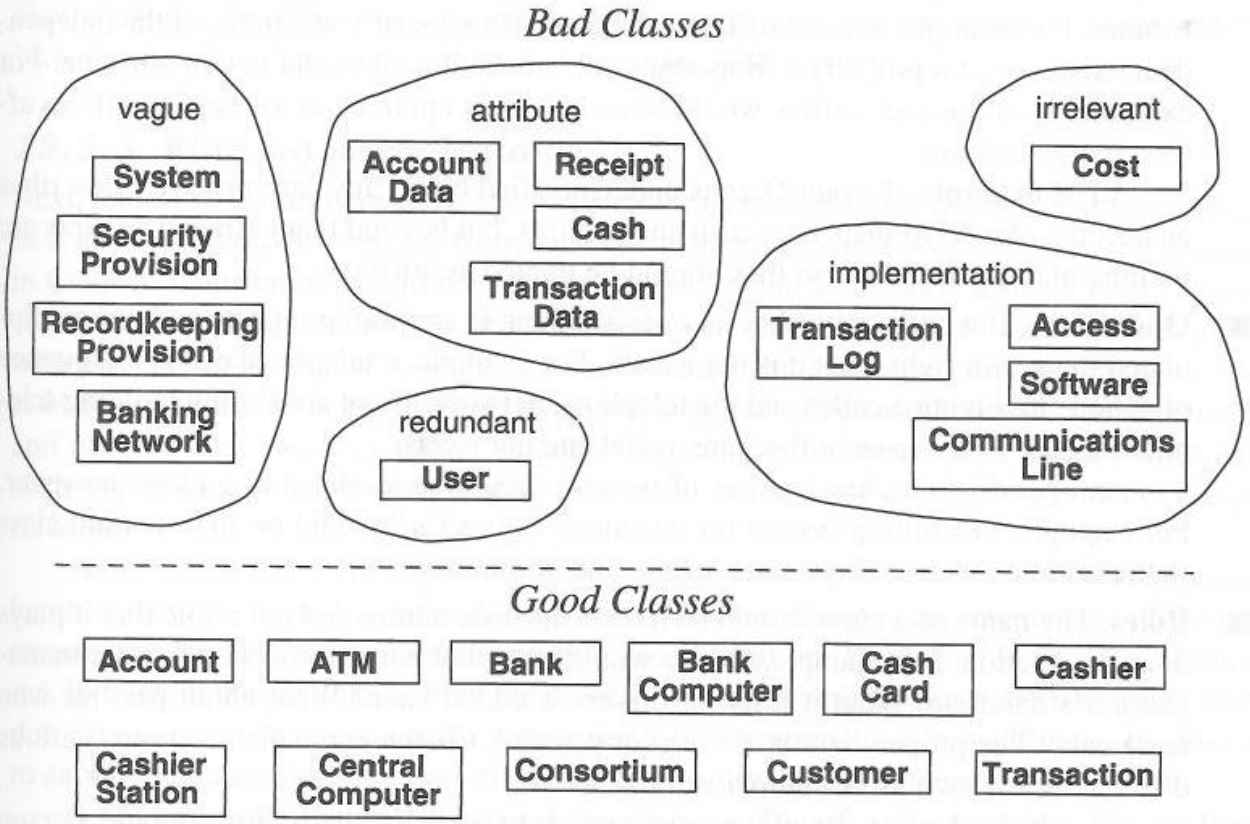
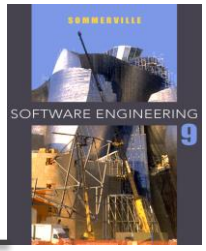


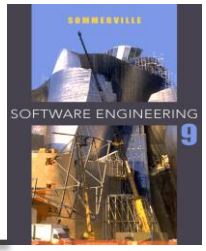
Figure 12.5 Eliminating unnecessary classes from ATM problem

Approaches to identification



- Use a grammatical approach based on a natural language description of the system (used in Hood OOD method).
- Base the identification on tangible things in the application domain.
- Use a behavioural approach and identify objects based on what participates in what behaviour.
- Use a scenario-based analysis. The objects, attributes and methods in each scenario are identified.

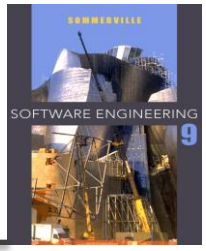
Weather station description



A weather station is a package of software controlled instruments which collects data, performs some data processing and transmits this data for further processing. The instruments include air and ground thermometers, an anemometer, a wind vane, a barometer and a rain gauge. Data is collected periodically.

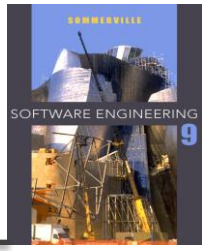
When a command is issued to transmit the weather data, the weather station processes and summarises the collected data. The summarised data is transmitted to the mapping computer when a request is received.

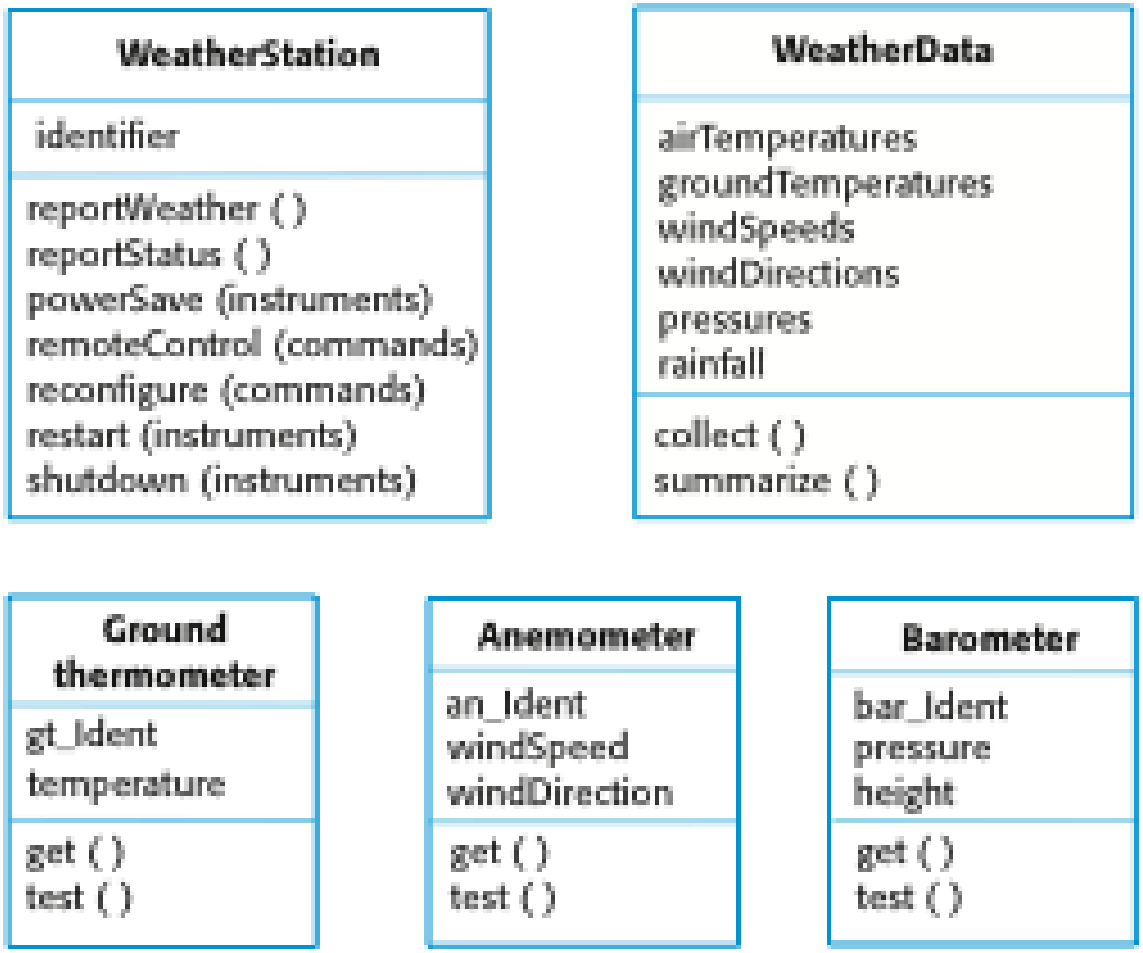
Weather station object classes

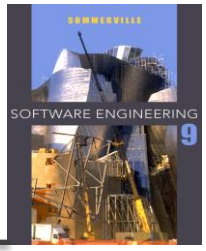


- Object class identification in the weather station system may be based on the tangible hardware and data in the system:
  - Ground thermometer, Anemometer, Barometer
    - Application domain objects that are ‘hardware’ objects related to the instruments in the system.
  - Weather station
    - The basic interface of the weather station to its environment. It therefore reflects the interactions identified in the use-case model.
  - Weather data
    - Encapsulates the summarized data from the instruments.

Weather station object classes

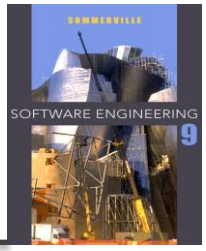






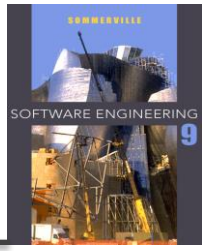
“For all the faculties of the CSE, department needs to allocate subjects based on Previous semester academic Feedback & Subject preference given by the faculty of their area of expertise. If the faculty has got less feedback in particular subject, then allocation of a subject shall produce warning message being issued by System”

Design models



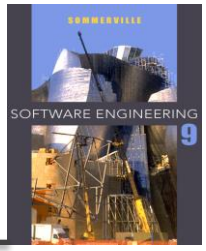
- Design models show the objects and object classes and relationships between these entities.
- Static models describe the static structure of the system in terms of object classes and relationships.
- Dynamic models describe the dynamic interactions between objects.

Examples of design models



- Subsystem models that show logical groupings of objects into coherent subsystems.
- Sequence models that show the sequence of object interactions.
- State machine models that show how individual objects change their state in response to events.

Subsystem models



- Shows how the design is organised into logically related groups of objects.
- In the UML, these are shown using packages - an encapsulation construct. This is a logical model. The actual organisation of objects in the system may be different.

Sequence diagrams

- Sequence diagrams are part of the UML and are used to model the interactions between the actors and the objects within a system.
- A sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance.
- The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these.
- Interactions between objects are indicated by annotated arrows.

Sequence Diagrams

- A sequence diagram shows the participants in an interaction and the sequence of messages among them
- A sequence diagram shows the interaction of a system with its actors to perform a use case

Sequence Diagrams (Asynchronous)

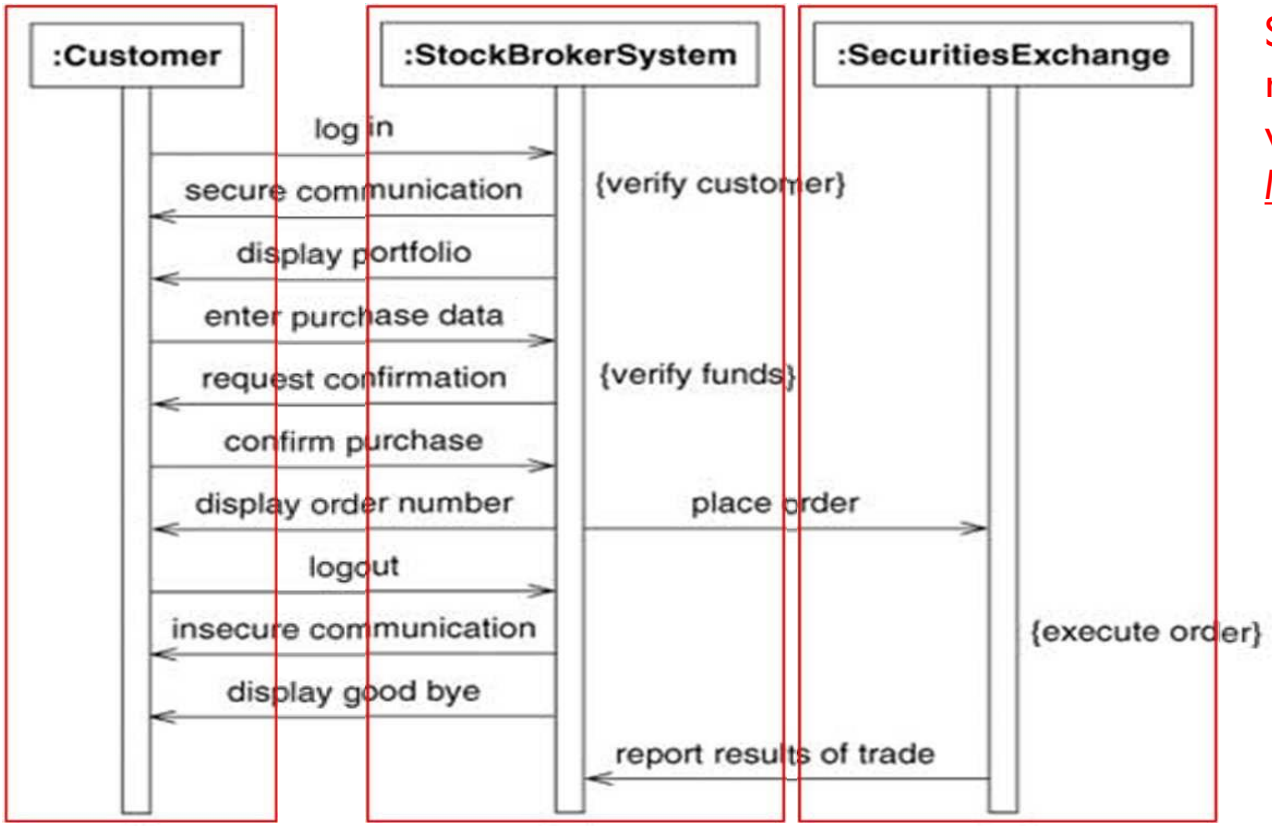


Figure 7.5 Sequence diagram for a session with an online stock broker. A sequence diagram shows the participants in an interaction and the sequence of messages among them.

Actors as well as the System are represented by a vertical line called a *lifeline*

Sequence Diagrams (Asynchronous)

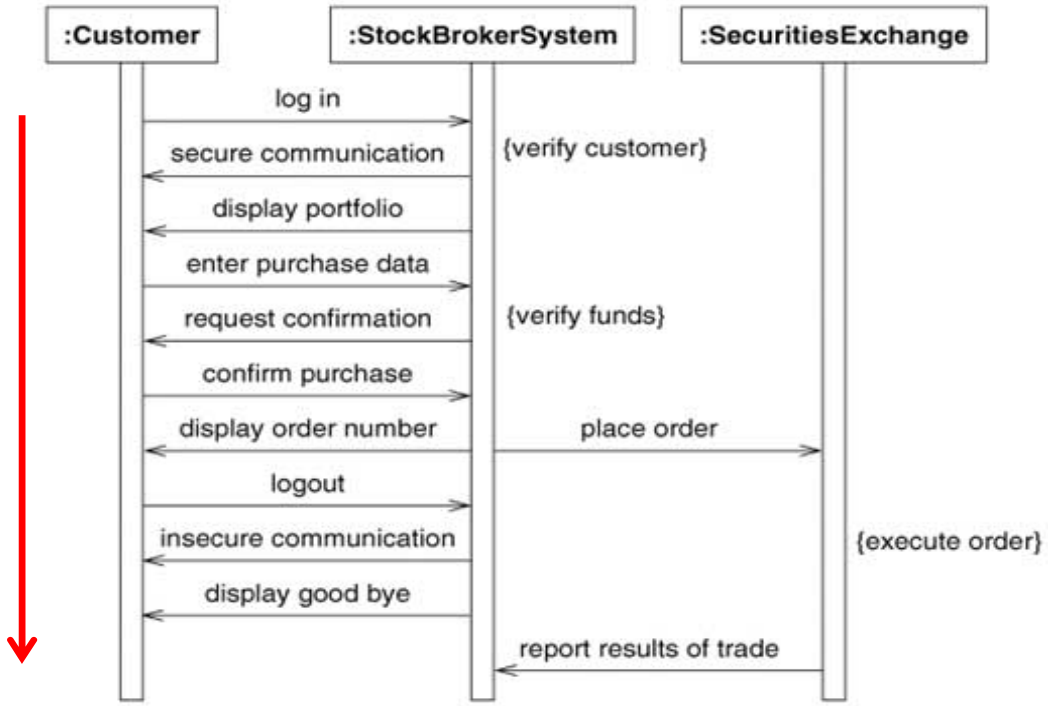


Figure 7.5 Sequence diagram for a session with an online stock broker. A sequence diagram shows the participants in an interaction and the sequence of messages among them.

Time proceeds from top to bottom but spacing is irrelevant

Sequence Diagrams (Asynchronous)

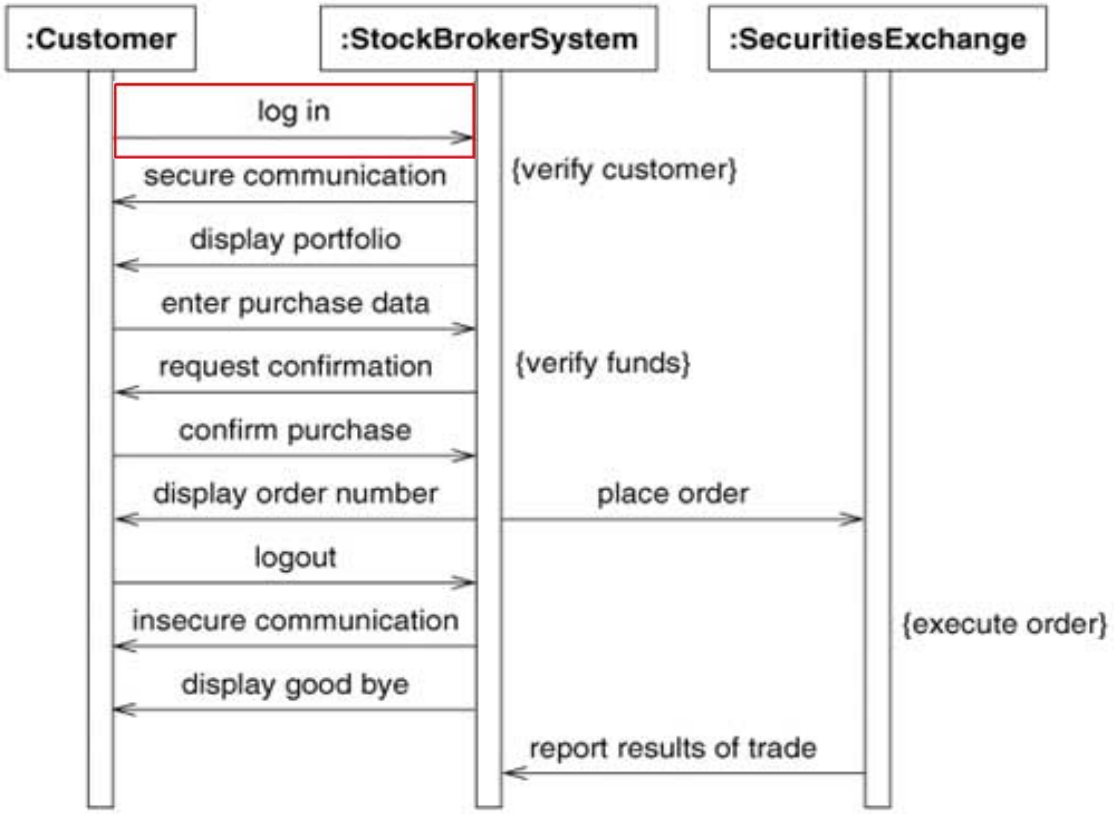


Figure 7.5 Sequence diagram for a session with an online stock broker. A sequence diagram shows the participants in an interaction and the sequence of messages among them.

Each message is a horizontal arrow from the sender to the receiver

Sequence Diagrams (Asynchronous)

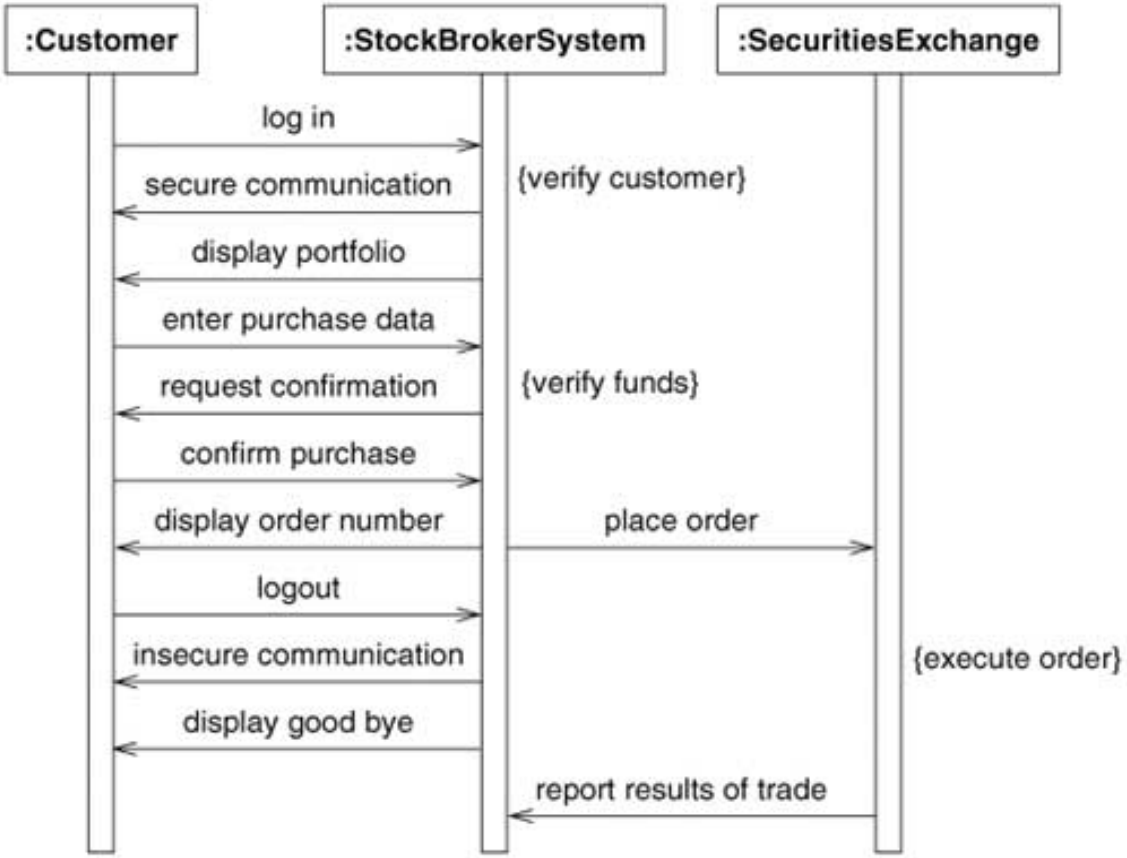


Figure 7.5 Sequence diagram for a session with an online stock broker. A sequence diagram shows the participants in an interaction and the sequence of messages among them.

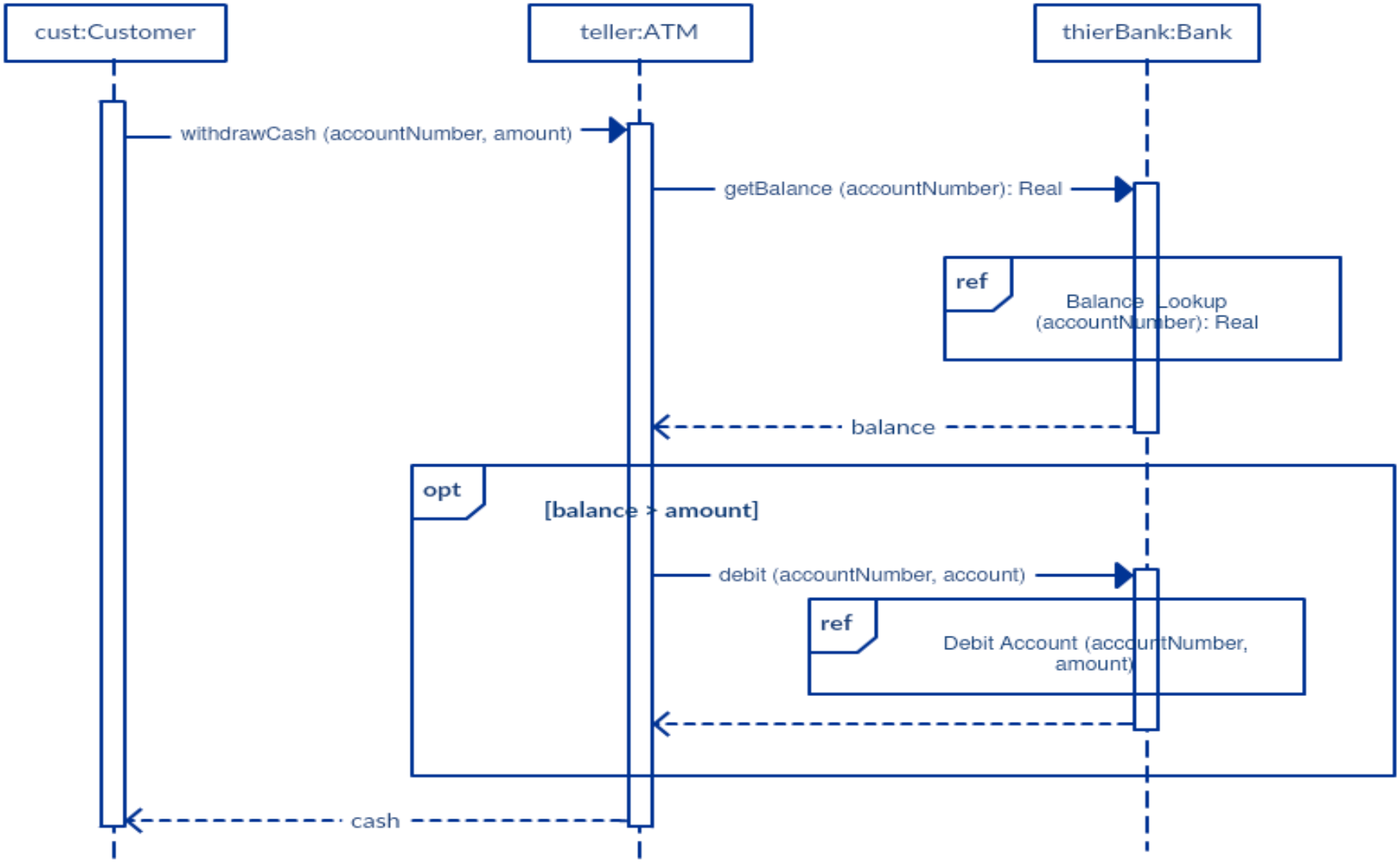
Are used for high level interactions

Can contain concurrent signals

- Stock Broker System sends messages to Customer and Securities Exchange concurrently

Signals between participants do not need to alternate

- Stock Broker System sends "secure communication" followed by "display portfolio"


cust:Customer
teller:ATM
thierBank:Bank
withdrawCash (accountNumber, amount)
getBalance (accountNumber): Real
ref
Balance Lookup
(accountNumber): Real
balance
opt
[balance > amount]
debit (accountNumber, account)
ref
Debit Account (accountNumber,
amount)
cash

State Modelling

- A state model describes the sequences of operations that occur in response to external stimuli.
- As opposed to what the operations do, what they operate on, or how they are implemented.
- Changes to objects and their relationships over time should be examined.
- A state model consists of multiple state diagrams, one for each class with temporal behavior that is important to an application.
- A state diagram relates events and states.
- Events represent external stimuli
- States represent values of (attributes) object.

States

- A state is an abstraction of the values and the links of an object.
- Sets of values and links are grouped together into a state according to the gross behavior of objects.
- Example: The state of an bank is either solvent or insolvent, depending on whether its assets exceed its liabilities.
- States often correspond to:
  - verbs with a suffix ‘ing’: waiting, dialing..
  - Or the duration of some condition: powered, below freezing.

States

- Notation:

Figure 5.4 States. A state is an abstraction of the values and links of an object.

- The objects in a class have a finite number of possible states. One or possibly larger number.
- Each object can only be in one state at time. Objects may parade through one to more states in their lifetime
- A state specifies the response of an object to input events.
- The response may include the invocation of a behavior or a change of state.

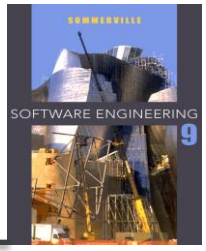
Transitions and Conditions

- A transition is an instantaneous change from one state to another.
- The transition is said to **fire** upon the change from the source state to the target state. Usually the origin and target states are different but may be the same.
- A transition occurs when its event occurs, unless an optional guard condition causes the event to be ignored.
- A guard condition is a boolean expression that must be true in order for a transition to occur.
- The choice of next state depends on both the source state and the event received.

State Diagram

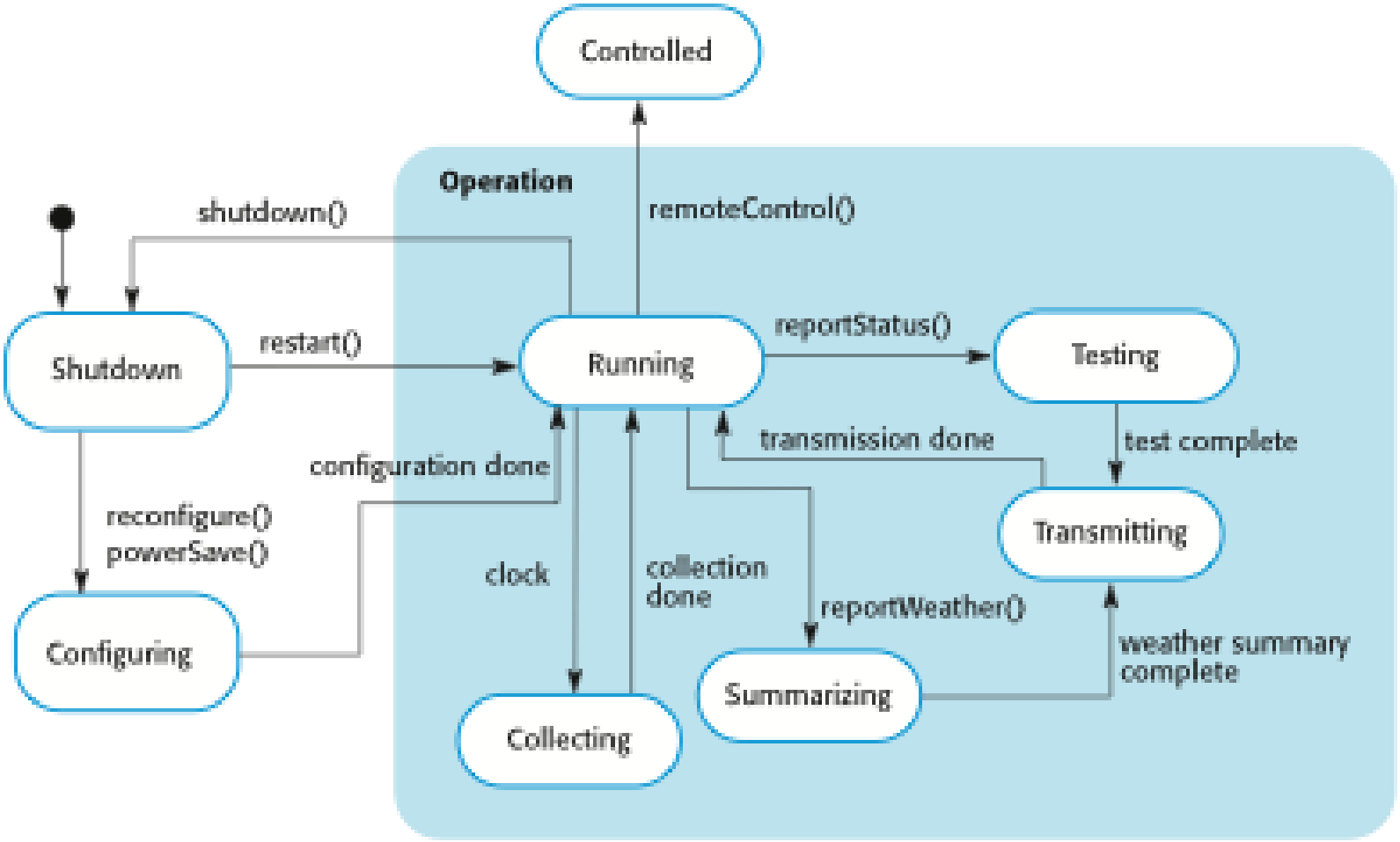
- A state diagram is a graph whose nodes are states and whose directed arcs are transitions between states.
- A state diagram specifies or describes the state sequence caused by event sequences.
- State names must be unique within the scope of the state diagram.
- All objects in a class execute the state diagram for that class.
- The class state diagram models the common behavior of the class objects.

State diagrams



- State diagrams are used to show how objects respond to different service requests and the state transitions triggered by these requests.
- State diagrams are useful high-level models of a system or an object's run-time behavior.
- You don't usually need a state diagram for all of the objects in the system. Many of the objects in a system are relatively simple and a state model adds unnecessary detail to the design.

Weather station state diagram



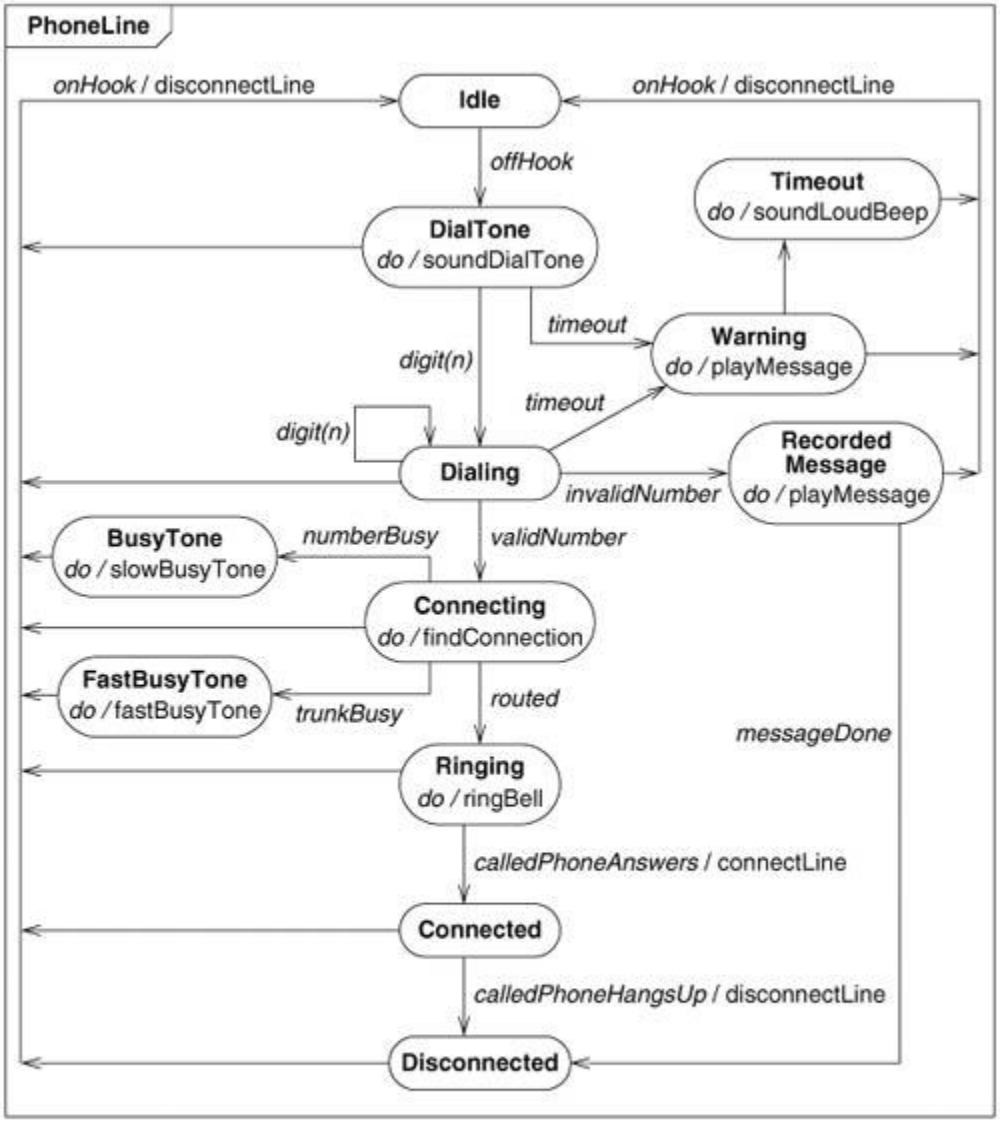
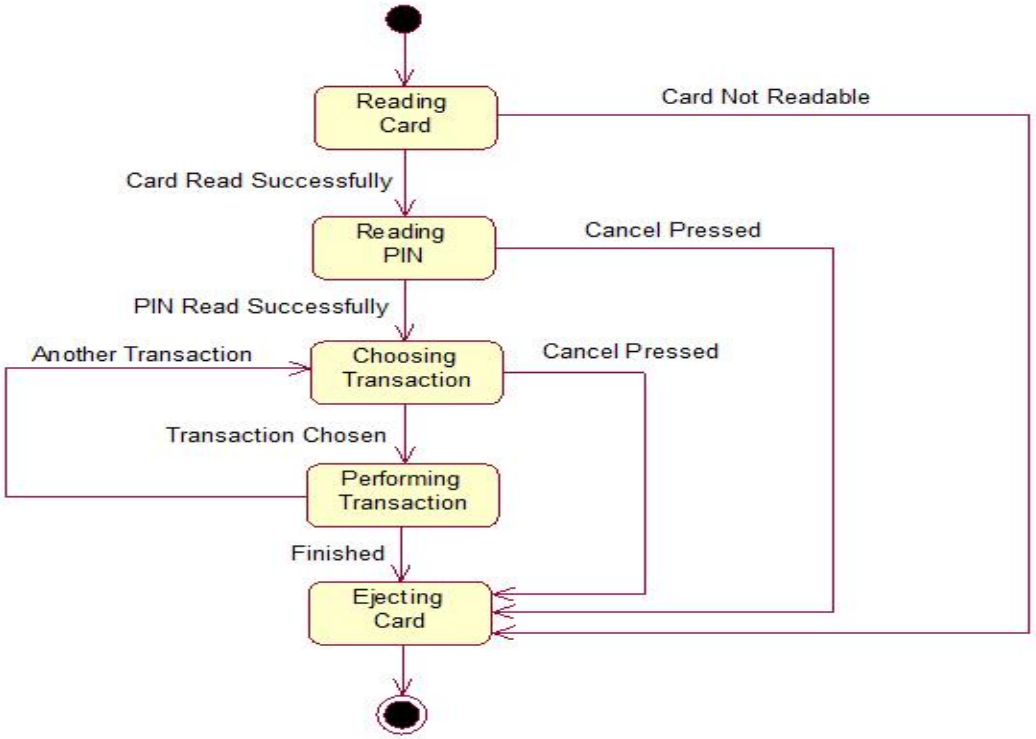
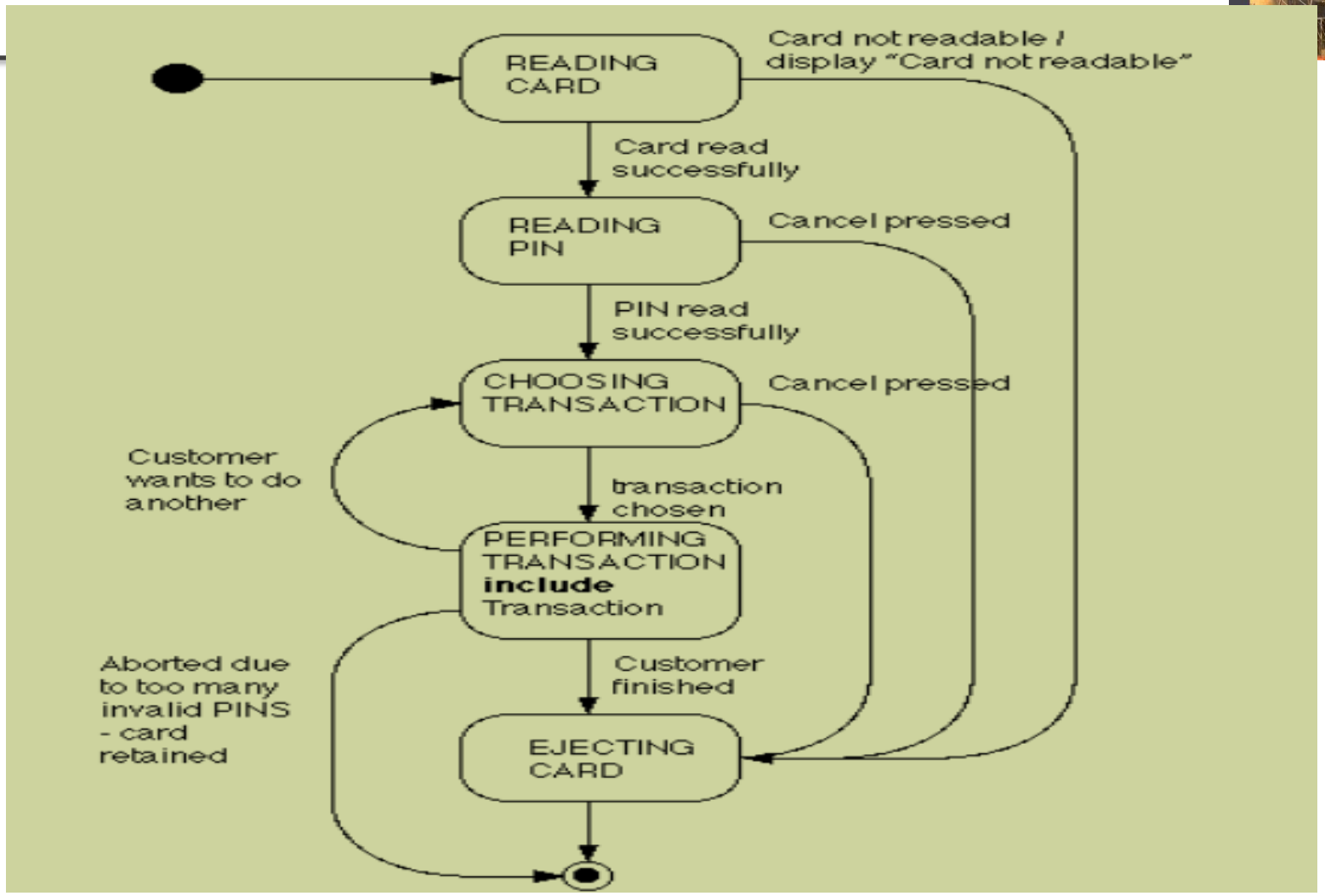


Figure 5.17 State diagram for phone line with activities. State diagrams let you express what objects do in response to events.

ATM Machine


Reading Card
Card Not Readable
Card Read Successfully
Reading PIN
Cancel Pressed
PIN Read Successfully
Another Transaction
Choosing Transaction
Cancel Pressed
Transaction Chosen
Performing Transaction
Finished
Ejecting Card

ATM STATE DIAGRAM



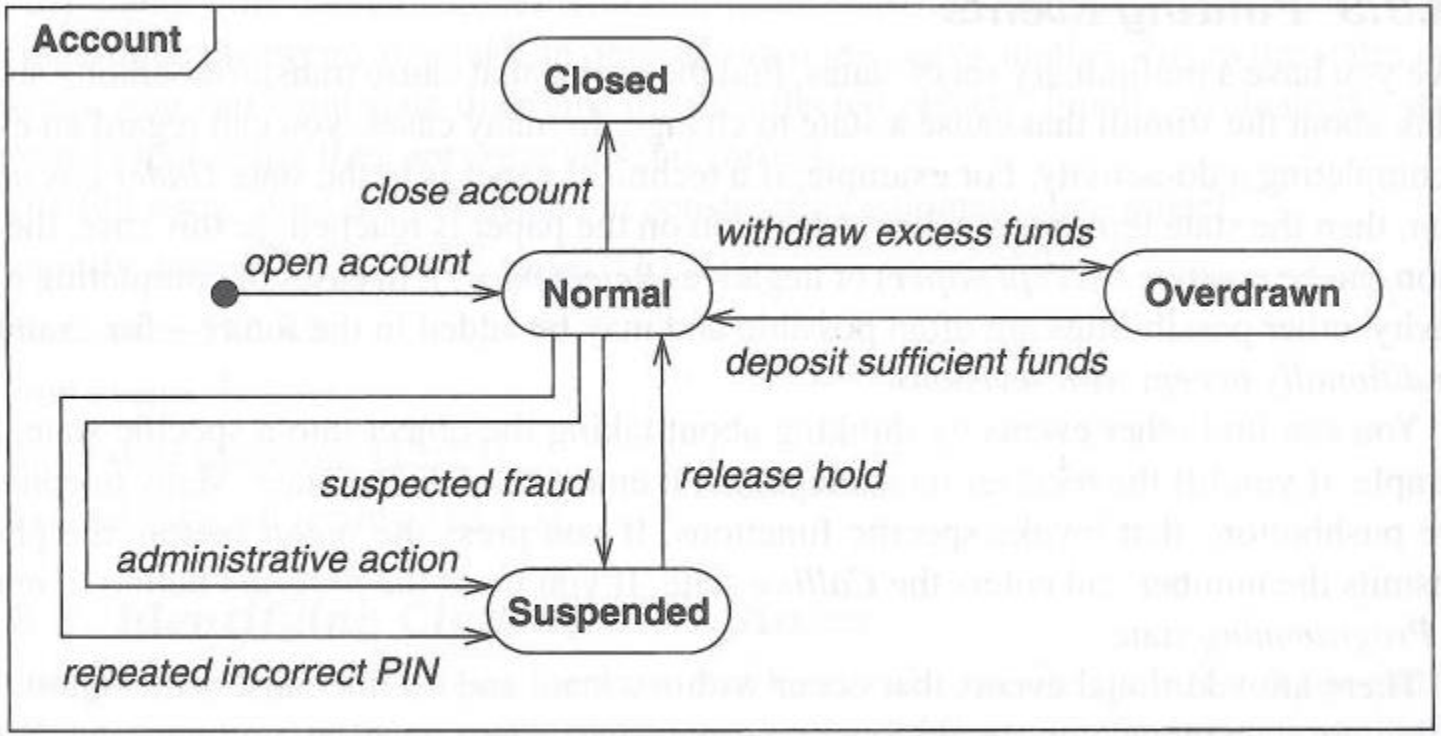
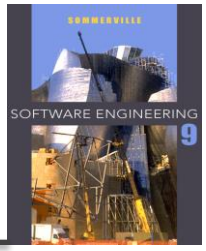


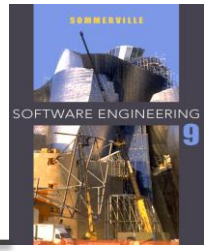
Figure 12.14 Domain state model. The domain state model documents important classes that change state in the real world.

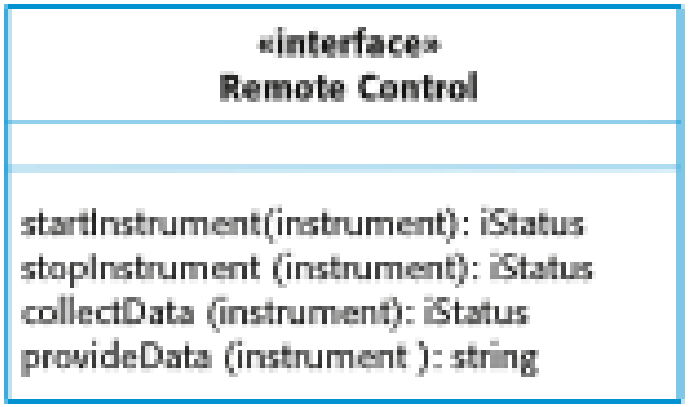
Interface specification

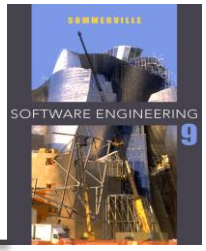


- Object interfaces have to be specified so that the objects and other components can be designed in parallel.
- Designers should avoid designing the interface representation but should hide this in the object itself.
- Objects may have several interfaces which are viewpoints on the methods provided.
- The UML uses class diagrams for interface specification but Java may also be used.

Weather station interfaces







Thank you