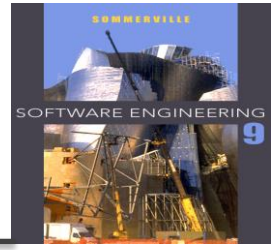
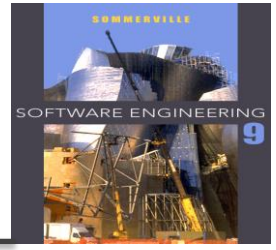


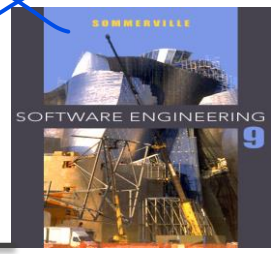
Chapter 3 – System Modeling and Architectural Design

Topics covered



- ✧ System Modeling
- ✧ UML Diagram Types
- ✧ Graphical Models
- ✧ Software Architecture
- ✧ Architectural Design
- ✧ Architectural Views
- ✧ Application Architectures
- ✧ Transaction Processing Systems

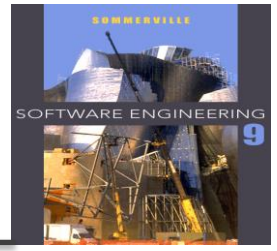
3.1 System Modeling



- 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.

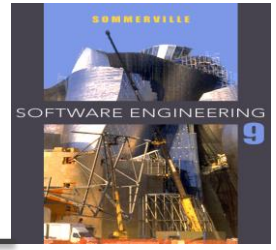
3.2 UML Diagram Types

Self (mcq)



- Activity diagrams, which show the activities involved in a process or in data processing .
- Use case diagrams, which show the interactions between a system and its environment.
- Sequence diagrams, which show interactions between actors and the system and between system components.
- Class diagrams, which show the object classes in the system and the associations between these classes.
- State diagrams, which show how the system reacts to internal and external events.

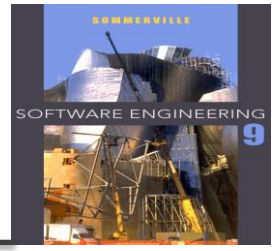
3.3 Graphical Models



1. Context Model: in this model, context diagram and activity diagram is used.
2. Interaction Model: in this model, use-case diagram and sequence diagram is used.
3. Structural Model: in this model, class diagram and object diagram is used.

3.3.1 Context Models

Self (mcq)



- Context models are used to illustrate the operational context of a system - they show what lies outside the system boundaries.
- Social and organisational concerns may affect the decision on where to position system boundaries.
- Architectural models show the system and its relationship with other systems.

Context Diagram {Not UML Diagram} for Android Blood-Bank and its Boundaries

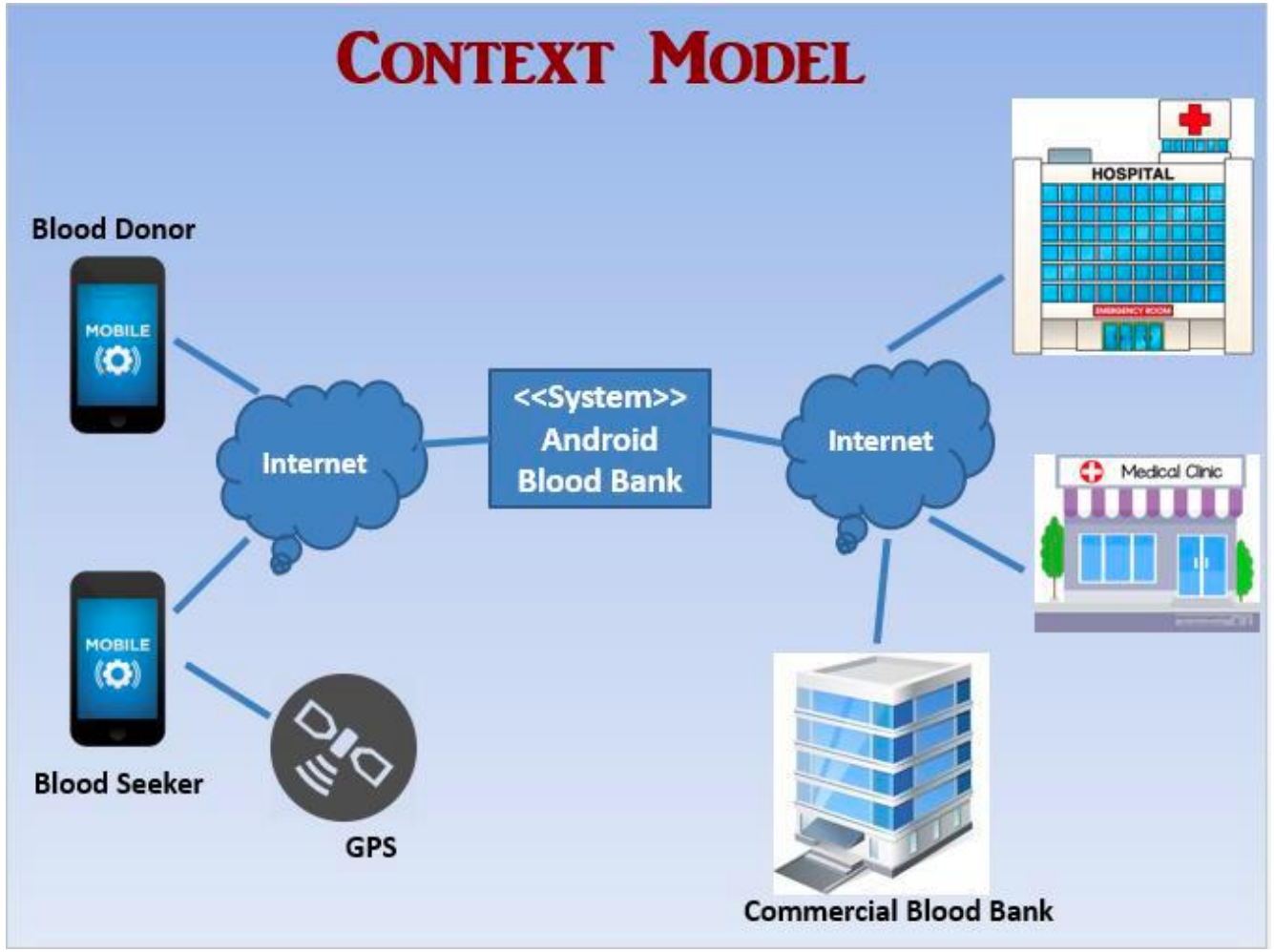
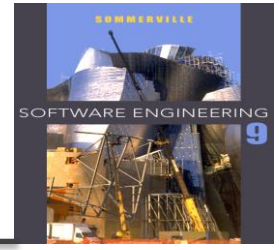


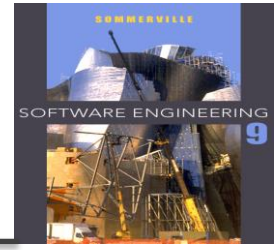
Figure 3.1: Context Diagram for Android Blood-Bank and its Boundaries

Context Models



- Activity diagrams are intended to show the activities that make up a system process and the flow of control from one activity to another.
- The start of a process is indicated by a filled circle; the end by a filled circle inside another circle.
- Rectangles with round corners represent activities, that is, the specific sub-processes that must be carried out.
- You may include objects in activity charts.
- UML activity diagrams are used to define major business process models.

UML Activity Diagram for Inventory System



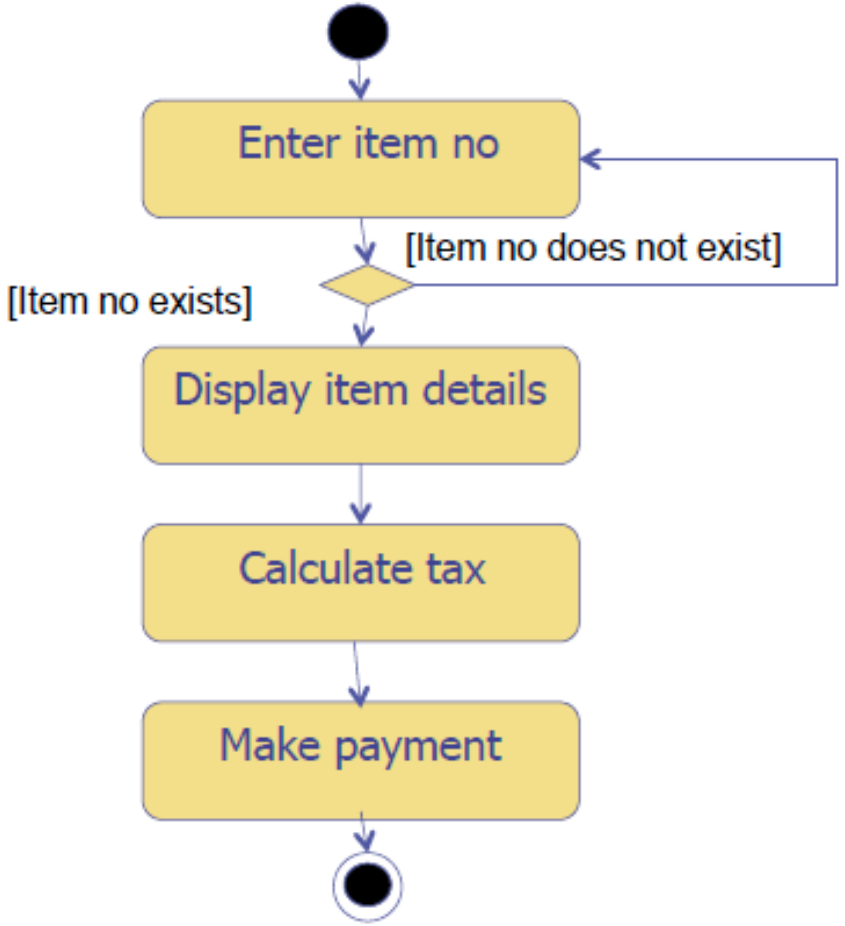
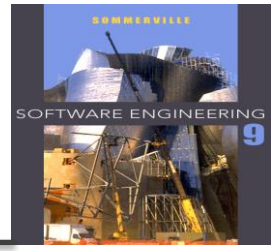


Figure 3.2: UML Activity Diagram for Inventory System

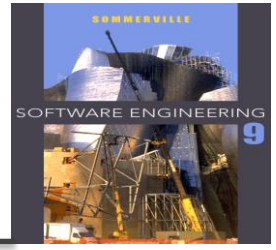
3.3.2 Interaction Models

Self (mcq)



- Modeling user interaction is important as it helps to identify user requirements.
- Modeling system-to-system interaction highlights the communication problems that may arise.
- Modeling component interaction helps us understand if a proposed system structure is likely to deliver the required system performance and dependability.
- Use case diagrams and sequence diagrams may be used for interaction modeling.

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.
- Use case diagrams give a fairly simple overview of an interaction so you have to provide more detail to understand what is involved. This detail can either be a simple textual description, a structured description in a table, or a sequence diagram.

Use-Case Diagram for ATM

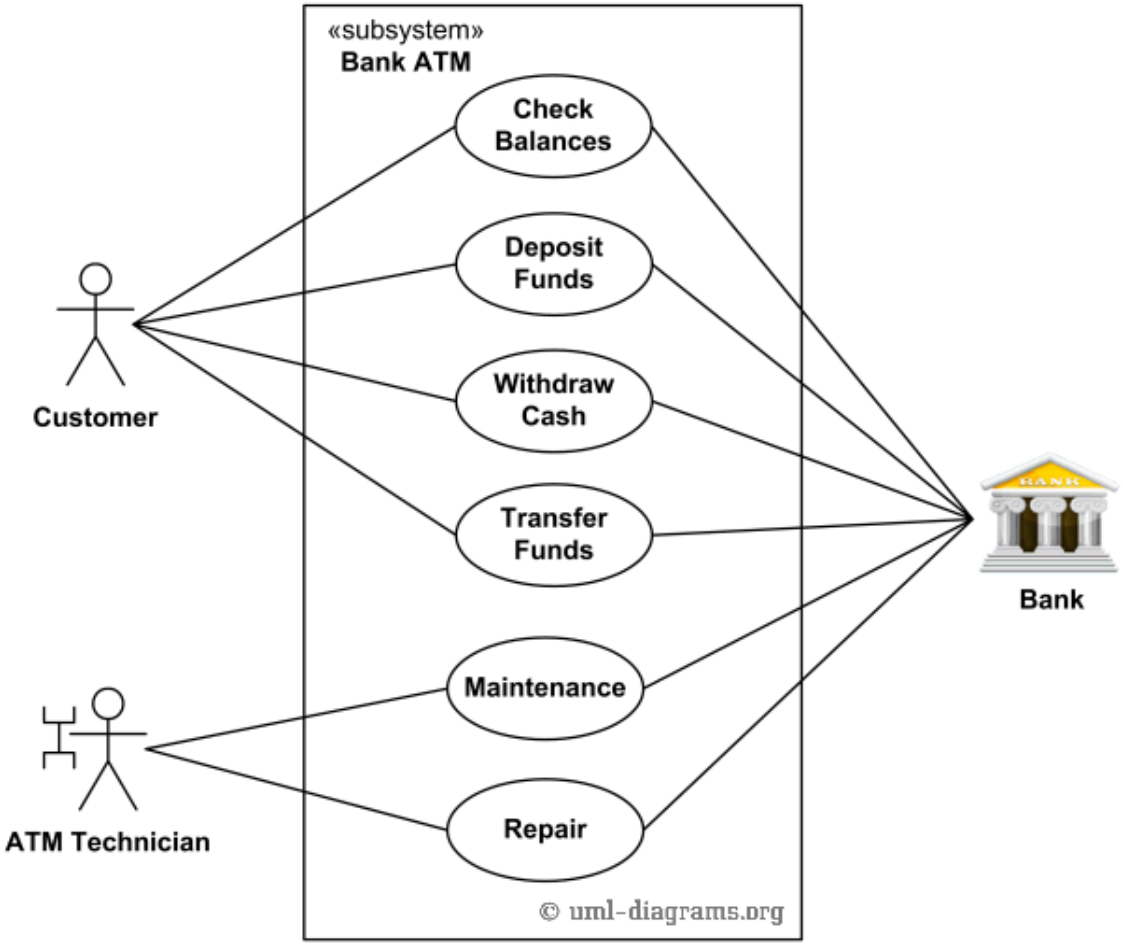
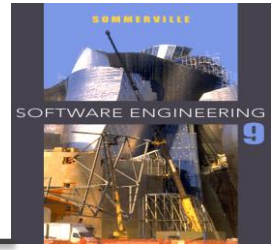


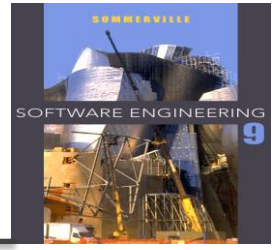
Figure 3.3: Use-Case Diagram for ATM

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 Diagram for ATM



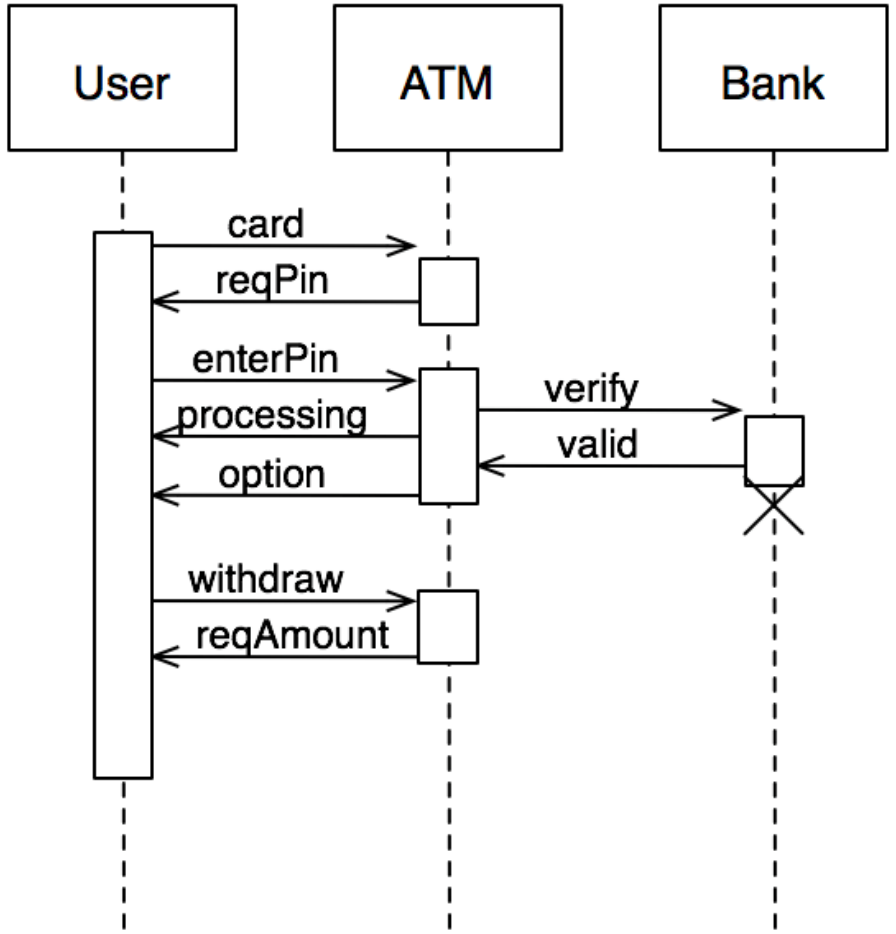
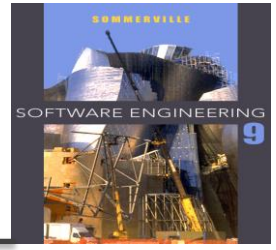


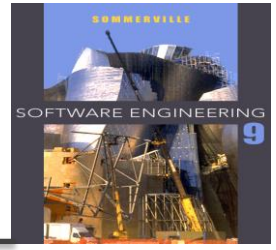
Figure 3.4: Sequence Diagram for ATM

3.3.3 Structural Models

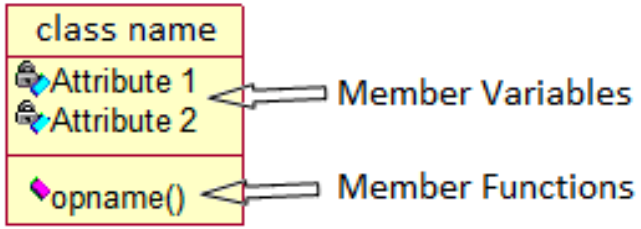


- Structural models of software display the organization of a system in terms of the components that make up that system and their relationships.
- Structural models may be static models, which show the structure of the system design, or dynamic models, which show the organization of the system when it is executing.
- You create structural models of a system when you are discussing and designing the system architecture.

Class Diagrams

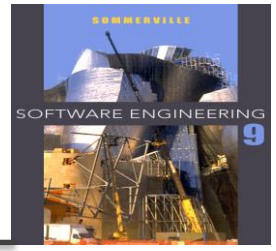


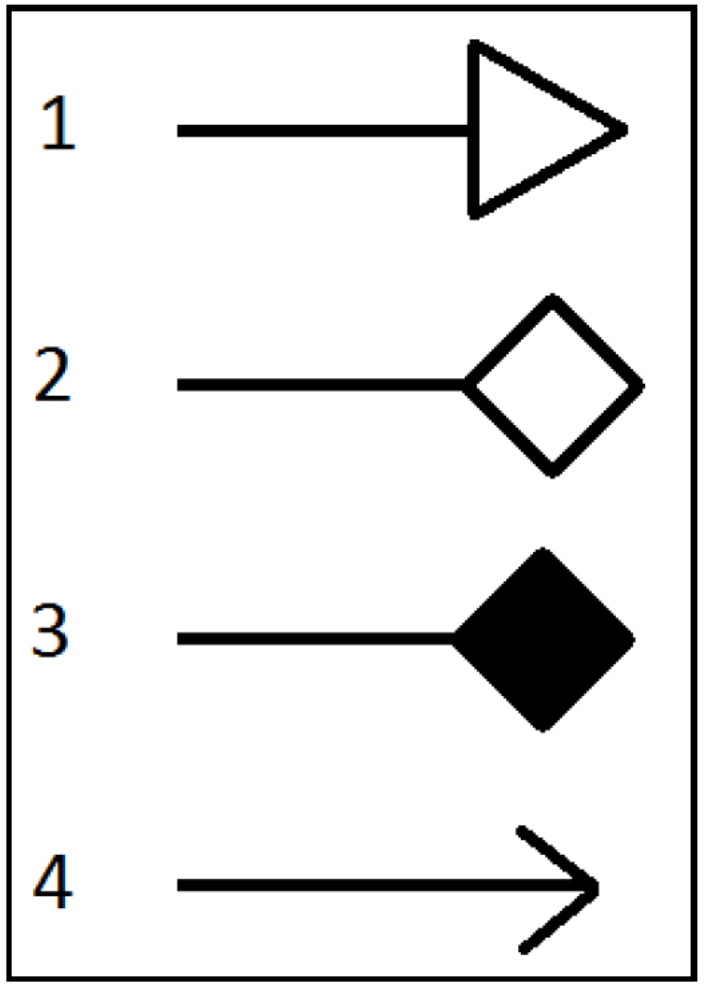
- Class diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between these classes.
- Class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML).
- **Class Diagram Notations:** UML class is represented by the diagram shown below that is divided into three sections.
  - First section is used to name the class.
  - Second section is used to show the attributes / member variable of the class.
  - Third section is used to describe the operations performed by the class.




```
Class Students {
private:
   int ID;
   char * Name;
public:
   void Registration();
};
```

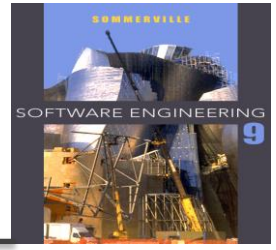
Types of Relationship Between the Classes





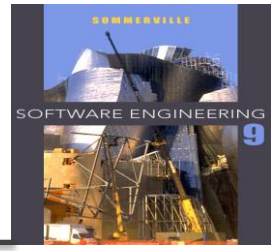
1. Shows **Generalization** between two classes to maintain parent-child relationship. (***is-a relation***)
2. Shows **Aggregation** between two classes to maintain containership. (***has-a relation***)
3. Shows **Composition** between two classes to maintain containership. (***has-a relation***)
4. Shows **Association** between two classes to maintain a simple connectivity.

Types of Relationship Between the Classes



- **Difference between Aggregation and Composition: (Has-a Relationship)**
  - In **aggregation** contained class exist even if main class is deleted. Eg: **Car has Engine**, If car is broken still engine exist and can be used in another car.
  - In **composition** contained class live or die on the basis of main class. Eg: **Folder has file**, If you delete a folder all files will be deleted as well.
- **Generalization: (Is-a Relationship)**
  - Generalization is used in class diagrams to deal with most powerful concept of object orientation that is **Inheritance**. Generalization is drawn between two classes to show Parent-Child relation.
  - In modeling systems, it is often useful to examine the classes in a system to see if there is scope for generalization. If changes are proposed, then you do not have to look at all classes in the system to see if they are affected by the change.
  - The lower-level classes are subclasses that inherit the attributes and operations from their superclass. These lower-level classes then add more specific attributes and operations.

Class Diagram Showing Different Type of Relationships



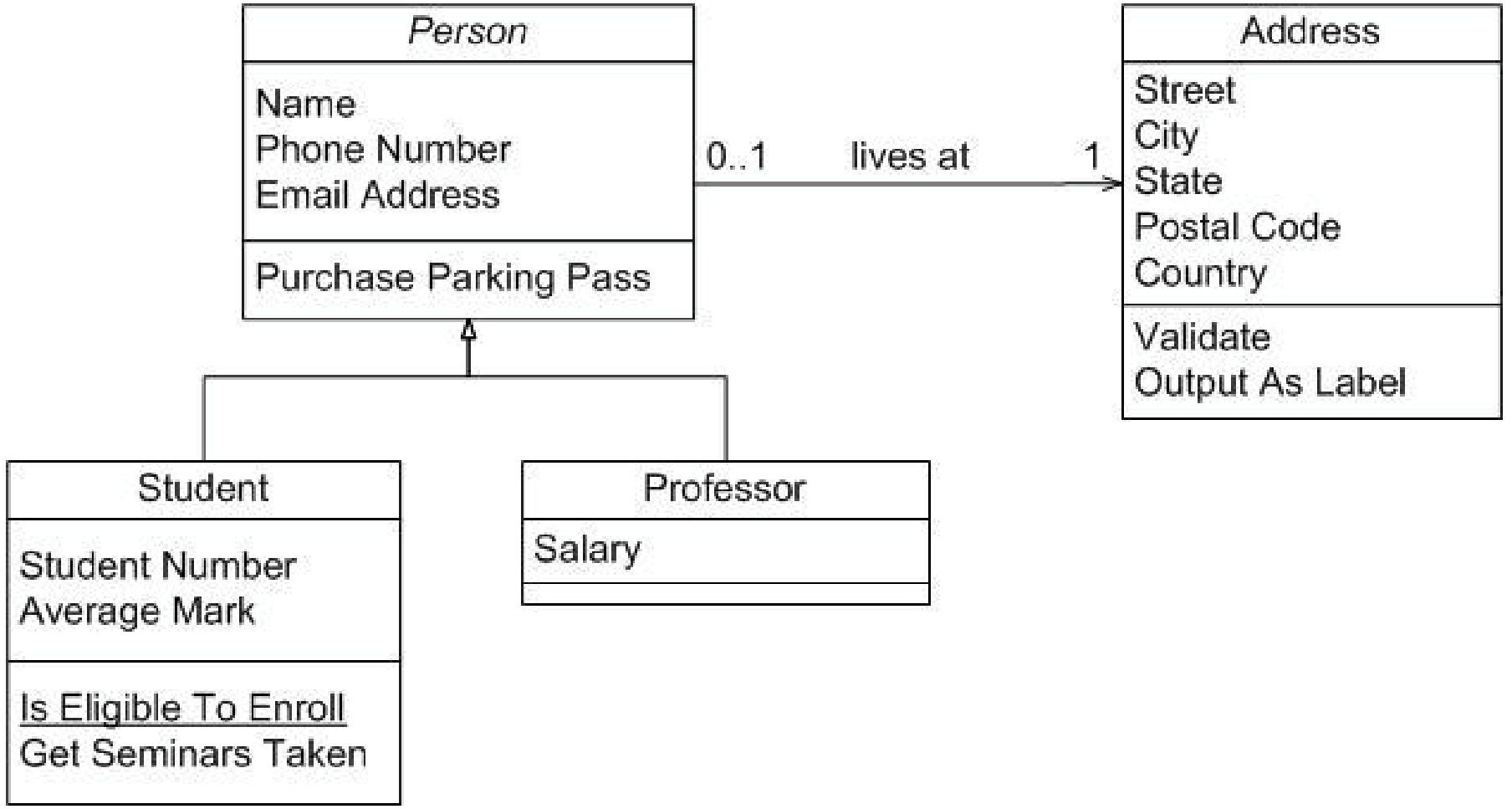
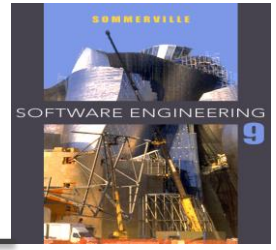


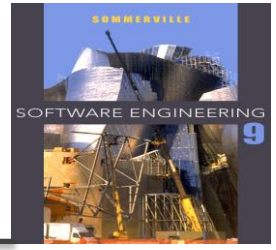
Figure3.5: Class Diagram Showing Different Type of Relationships

3.4 Architectural Design



- Architectural design is concerned with understanding how a software system should be organized and designing the overall structure of that system
- Architectural design is the first stage in the software design process.
- It is the critical link between design and requirements engineering, as it identifies the main structural components in a system and the relationships between them.
- .The output of the architectural design process is an architectural model that describes how the system is organized as a set of communicating components.

3.4.1 Use of Architectural Models



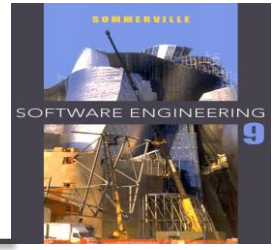
✧ As a way of facilitating discussion about the system design

- A high-level architectural view of a system is useful for communication with system stakeholders and project planning because it is not cluttered with detail. Stakeholders can relate to it and understand an abstract view of the system. They can then discuss the system as a whole without being confused by detail.

✧ As a way of documenting an architecture that has been designed

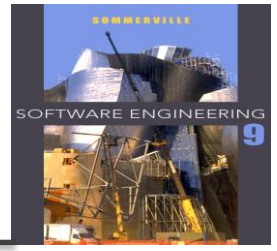
- The aim here is to produce a complete system model that shows the different components in a system, their interfaces and their connections.

3.5 Architectural Views



- Each architectural model only shows one view or perspective of the system.
  - It might show how a system is decomposed into modules, how the run-time processes interact or the different ways in which system components are distributed across a network. For both design and documentation, you usually need to present multiple views of the software architecture.
- Krutchen in his well-known 4 +1 view model of software architecture, suggests that there should be four fundamental architectural views, which can be linked through common use cases or scenarios (shown in Fig: 3.6 – Next slide)

4 + 1 View Model of Software Architecture



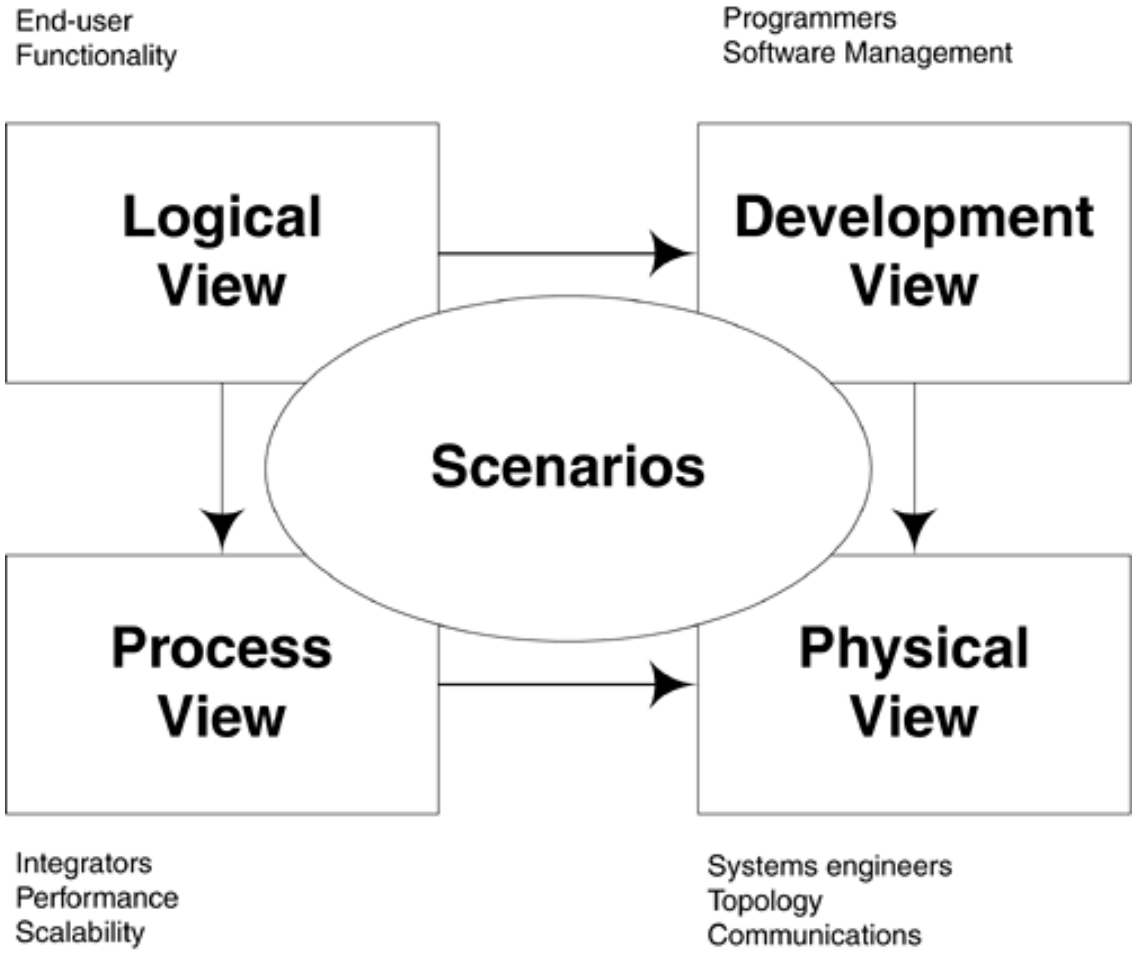
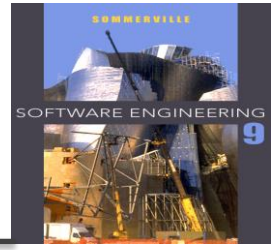


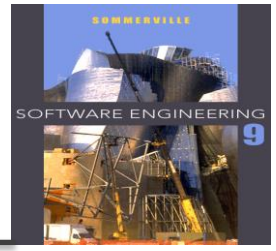
Figure 3.6: 4+1 View Model

4 + 1 View Model of Software Architecture



- **A logical view**, which shows the key abstractions in the system as objects or object classes.
- **A process view**, which shows how, at run-time, the system is composed of interacting processes.
- **A development view**, which shows how the software is decomposed for development.
- **A physical view**, which shows the system hardware and how software components are distributed across the processors in the system.
- Related using use cases or scenarios (+1)

3.6 Application Architectures



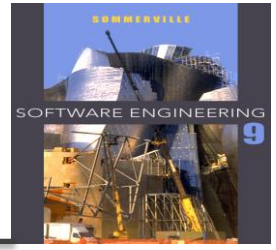
✧ Application systems are designed to meet an organizational need.

✧ As businesses have much in common, their application systems also tend to have a common architecture that reflects the application requirements.

Use of Application Architectures

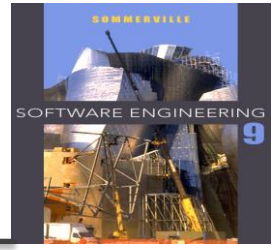
- As a starting point for architectural design.
- As a design checklist.
- As a way of organizing the work of the development team.
- As a means of assessing components for reuse.
- As a vocabulary for talking about application types.

3.7 Transaction Processing Systems (TPS)



- ✧ TPS is a type of information system that collects, stores, modifies and retrieves the data transactions of an organization.
  - For example - airline reservation systems, electronic transfer of funds, bank account processing systems.
- ✧ From a user perspective a transaction is:
  - Any coherent sequence of operations that satisfies a goal;
  - For example - find the times of flights from London to Paris.
- ✧ Users make asynchronous requests for service which are then processed by a transaction manager.

3.7.1 Application Architecture of TPS



- ✧ Transaction processing systems are usually interactive systems in which users make asynchronous requests for service.
  - Asynchronous means that you do not halt all other operations while waiting for the web service call to return.
- ✧ Figure 5.2 illustrates the conceptual architectural structure of TPS.
  - First a user makes a request to the system through an I/O processing component. The request is processed by some application specific logic. A transaction is created and passed to a transaction manager, which is usually embedded in the database management system. After the transaction manager has ensured that the transaction is properly completed, it signals to the application that processing has finished.

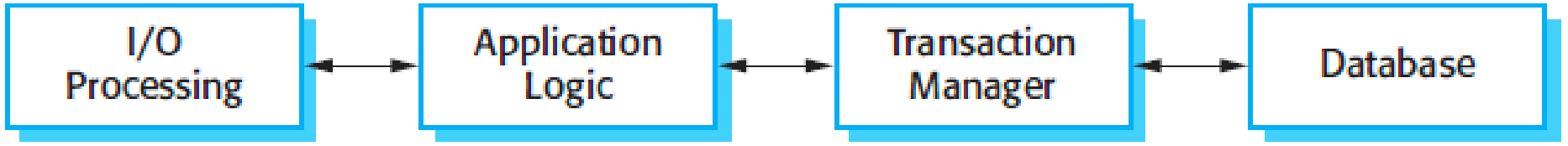
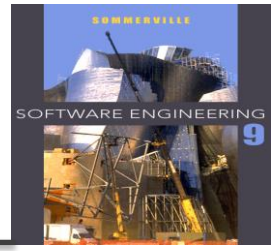


Figure 3.7: Architecture of TPS

3.7.2 The Software Architecture of an ATM System



- An example of a transaction is a customer request to withdraw money from a bank account using an ATM, Figure 5.3. This involves getting details of the customer's account, checking the balance, modifying the balance by the amount withdrawn, and sending commands to the ATM to deliver the cash. Until all of these steps have been completed, the transaction is incomplete and the customer accounts database is not changed.

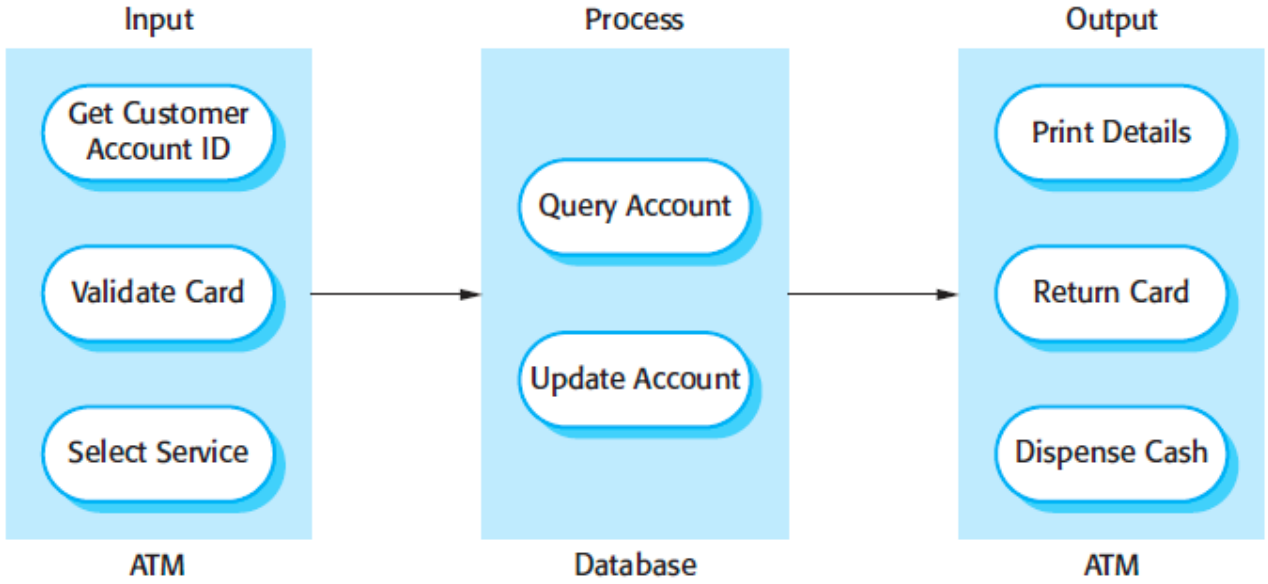
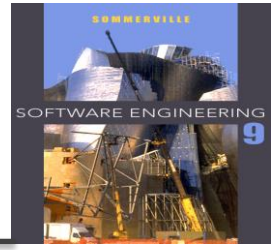


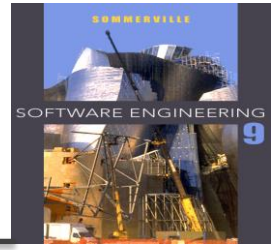
Figure 3.8: Architecture of ATM

Key Points



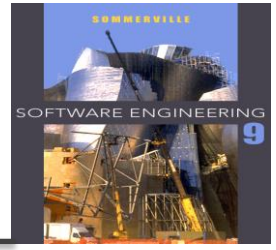
- A model is an abstract view of a system that ignores system details. Complementary system models can be developed to show the system's context, interactions, structure and behavior.
- Context models show how a system that is being modeled is positioned in an environment with other systems and processes.
- Use case diagrams and sequence diagrams are used to describe the interactions between users and systems in the system being designed. Use cases describe interactions between a system and external actors; sequence diagrams add more information to these by showing interactions between system objects.
- Structural models show the organization and architecture of a system. Class diagrams are used to define the static structure of classes in a system and their associations.

Key Points



- A software architecture is a description of how a software system is organized.
- Architectural design decisions include decisions on the type of application, the distribution of the system, the architectural styles to be used.
- Architectures may be documented from several different perspectives or views such as a conceptual view, a logical view, a process view, and a development view.
- TPS is a type of information system that collects, stores, modifies and retrieves the data transactions of an organization.

Review questions



1. Define: system modeling, Activity diagram, Use case diagram, sequence diagram, class diagram, state diagram
2. Explain various graphical models
3. Draw Use-case and Sequence diagram for ATM
4. Difference between Aggregation and Composition.
5. What is Architectural design? What are the uses of architectural model?
6. Illustrate 4+1 view model of software architecture with neat diagram
7. List down the uses of application architecture.
8. What is TPS? Explain application architecture of TPS with neat diagram.