Object Oriented Analysis and Design

Object Oriented Analysis and Design (OOAD)

- Introduction to Object Oriented Concepts
- Class, Attributes, Methods and Object
- Messages
- Abstraction, Encapsulation and Data Hiding
- Class Hierarchy, Inheritance and Polymorphism
- Relationships (Is-a, Has-a, Uses-a)
- Role of UML in OOAD
- Object Modeling Technique

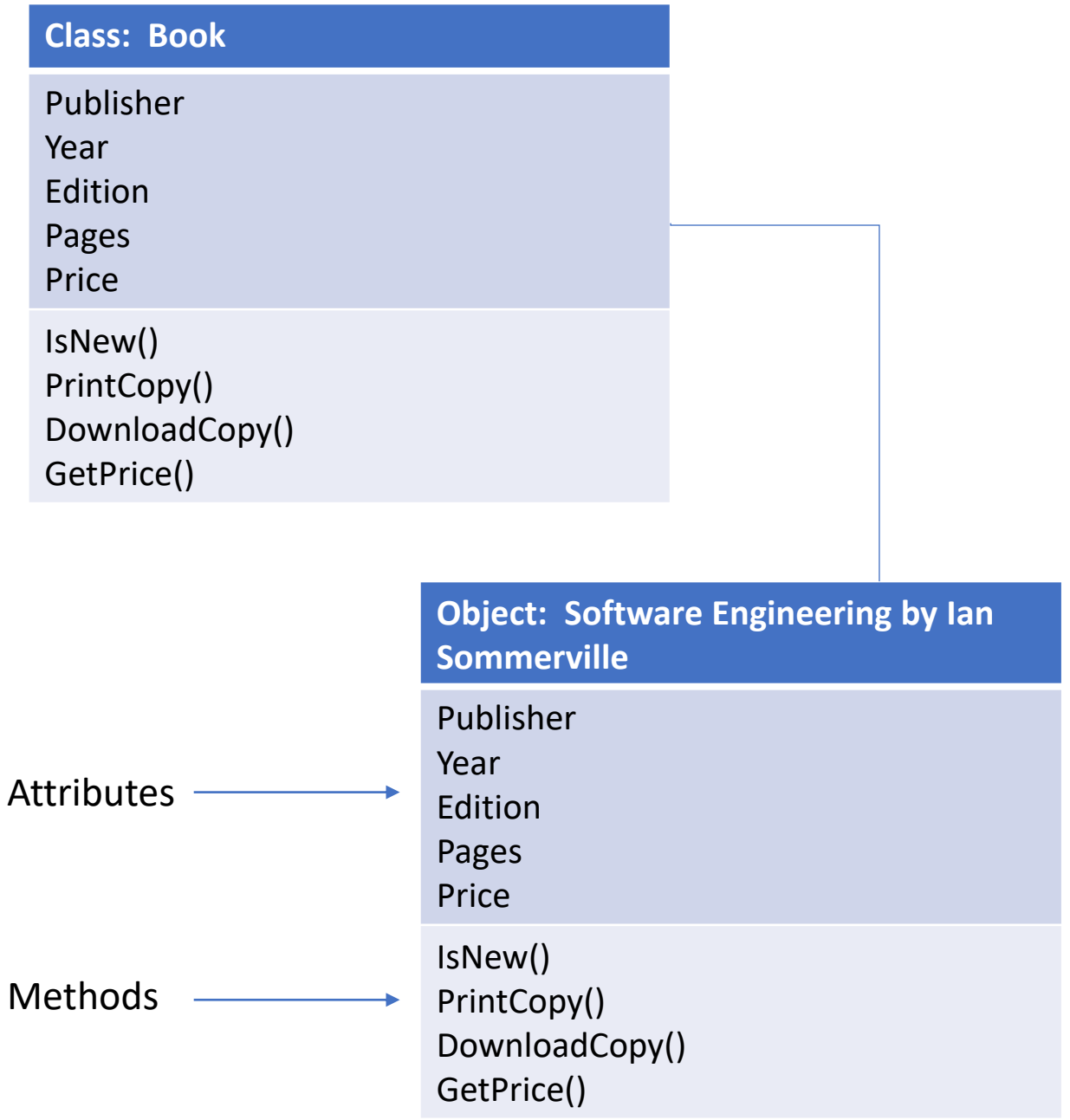
Object

- ..a concept, abstraction or thing with crisp boundaries and meaning for the problem in hand
- Purpose: 1. Promote understanding of the real world and 2. Provide a practical basis for computer implementations
- Decomposition of a problem into objects depends on judgment and the nature of the problem.

Object: Software Engineering by Ian Sommerville
Publisher Year Edition Pages Price
IsNew() PrintCopy() DownloadCopy()

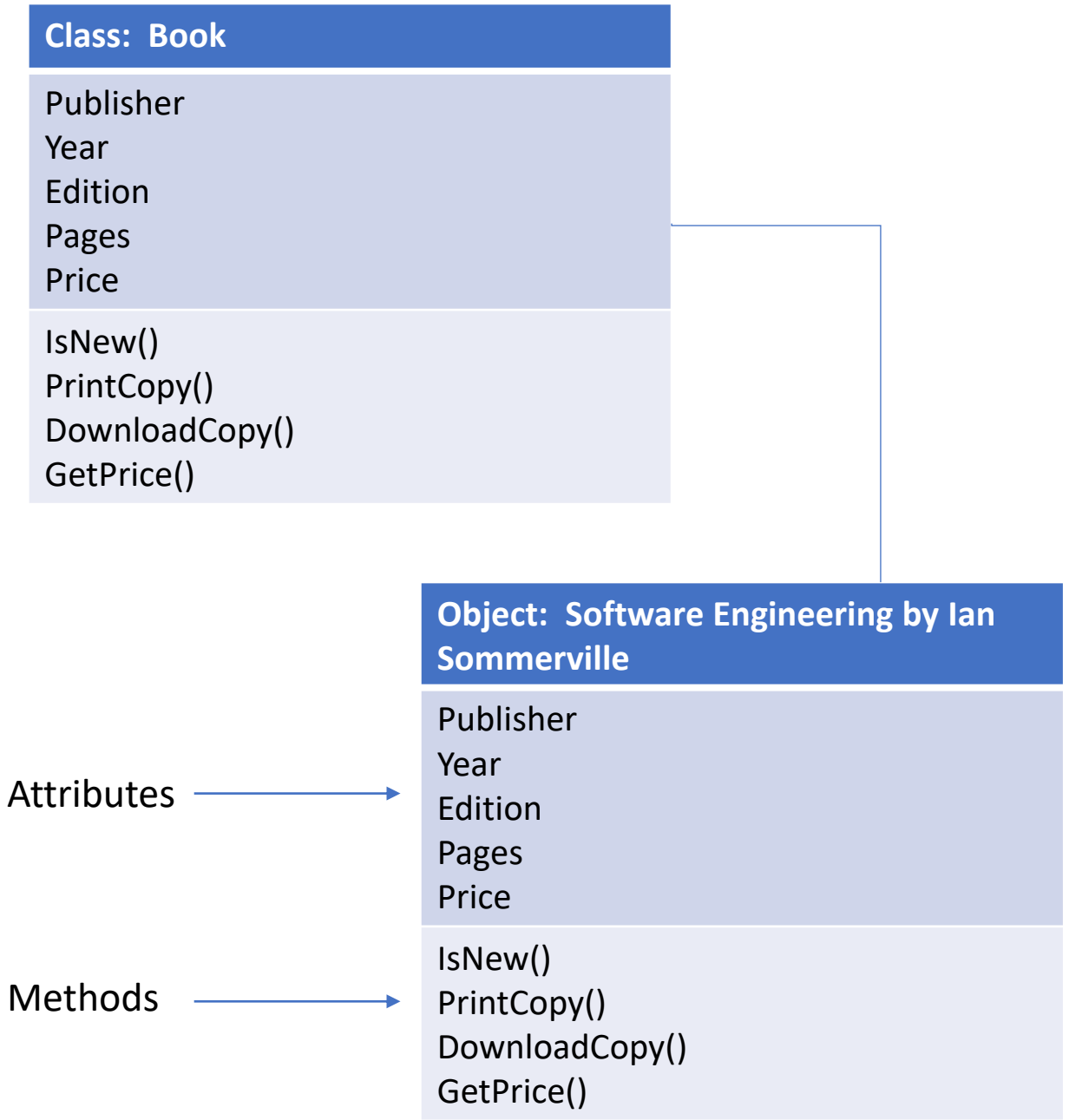
Class

- …describes a group of objects with similar properties(attributes), common behavior(operations), common relationships to other objects and common semantics. E.g.., *Person, Company, Animal*
- Each object “knows” it class
- By grouping objects into Classes, we abstract a problem(and hence obtain the power of abstraction)



Data Hiding

- Data in objects can be accessed only through methods
- Objects provide encapsulation of procedures and data – encapsulation provides info. Hiding
- Internal methods can be hidden

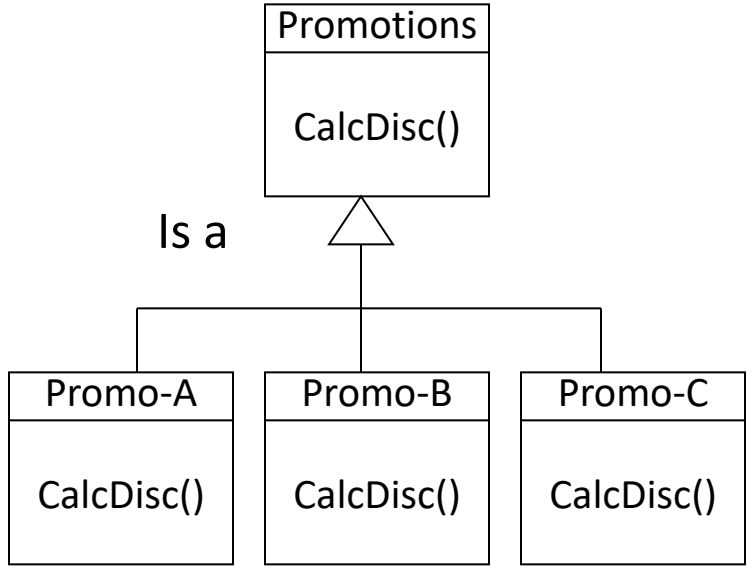


Book Rental Store

- Modern Book Rental Store(MBRS) rents books and it wants to automate its operations by managing the book inventory that has variety of books, and its customer data and the history of customers' transactions. MBRS announces various promotions such as 'Promo-A: 20% discount on Kids books' , 'Promo-B: 2 Kids books with zero rental fee if you rent 10 non-fiction books in 30 days', 'Promo-C: Rent 5 new books in a week and get zero rent on one book' and it has plans to announce more such promotions. In MBRS, for every customer who rents books, the transaction is calculated and optimized to get the minimum price and customer transactions are stored for future use.

Inheritance

- A powerful concept that makes it easy to show ‘inherited’ relationship (‘Is a’)
- Promo-A ‘Is a’ type of promotion
- The base class is ‘Promotions’
- It has CalcDisc() method to calculate discount
- Children can use the same method or override it by having their own implementation



Messages support polymorphism across objects (Adaptability)
Classes implement inheritance within class hierarchies. We can add a sub-class without inducing any changes in other classes (Extensibility).

Message

Message is an invocation of method in an object.

For example, when a customer rents a set of books, the system in the book rental store needs to calculate the rent and apply discount. Let us assume that we have a class called ‘Sale Optimizer’ for this purpose.

An object instantiated from ‘Sale Optimizer’ would invoke ‘CalcDisc()’ method.

Relationships

“Is-a” - Promo-A is a Promotion (inheritance)

‘Has-a’ (Aggregation) or ‘part of’. Example: A car has a wheel

‘Uses-a’ – The method of one class may use the object of another class. This is called ‘uses-a’ relationship. Example: A car uses petrol for functioning (petrol is an object of ‘fuel’ class).

Elements of Object Model

- Abstraction
- Encapsulation
- Modularity
- Hierarchy
- Typing (An object is an instance of a single class)
- Concurrency
- Persistence

Three Key Concepts

- Objects provide encapsulation of procedures and data – encapsulation provides info. hiding
- Messages that support polymorphism across objects - adaptability
- Classes that implement inheritance within class hierarchies - extensibility

Benefits

- Maintainable small chunks of code
- Extensibility
- Reusability
- Productivity
- OOT is enables Resilient, Simple, Approachable Software Architectures

OOAD Methodologies

- OOA, OOD – Codd and Yourdon 1991
- OMT from Rumbaugh et. al. 1991
- Object-Oriented Analysis and Design with Applications (OOADA) from Booch 1992
- Object-Oriented Analysis and Design (OOAD) from Martin 1993
- Object-Oriented Software Engineering (OOSE) from Jacobson et al. 1992

Question 1

Object Oriented Analysis and Design involves

A. Focusing on real world objects
B. Understanding the context and problem domain in order to identify actors and use cases
C. No mathematical foundation to arrive at analysis or design models
D. All of the above

ANSWER: D

Question 2

Which of the following is not associated with a class

A. Class name
B. Attributes
C. Database table definitions
D. Methods

ANSWER: C

Question 3

Which of the following is created when you instantiate Employee class only once?

A. An object of the class Employee
B. Sub-class of Employee
C. Different objects of Employee
D. Methods of Employee

ANSWER: A

Question 4

Which of the following is true?

A. A class can provide direct access on its attributes to other classes without using any methods
B. A method inherited from a super class will always have the same behavior in all sub-classes
C. An object instantiated can correspond to more than one class
D. Inheritance enables extensibility

ANSWER: D

Question 5

Which of the following is a OOAD methodology?

A. Object-Oriented Modeling Technique
B. Object Modeling Technique
C. Object-Oriented Methodology and Technique
D. Object-Oriented Spiral Methodology

ANSWER: B

OMT (Object Modeling Technique)

OMT divides Analysis and Design in three parts:

1. Analysis: Building of a model of the real-world situation starting with a problem statement.
2. System Design: Design of the overall architecture of the system.
3. Object Design: Refinement of the object structure towards efficient implementation, implementation details are added to the objects.

…..and implementation comes at the end.

OMT – The essence

- Identification and organization of application-domain concepts, instead of implementation domain concepts
- The OMT method has very rich notations. For the development of most systems only two/third of the possible notations will be used, but some systems need the more advanced modeling notations

Analysis

- Problem understanding
- Real world mapping of requirements
- Derives an abstraction of “what” the desired system must do (and not “how” to do)
- The objects in the analysis model should be application-domain concepts (not computer implementation concepts – data structures for eg.,)
- Users can understand and comment on a good analysis model

System Design

- Reflects high-level architecture
- Decomposition into sub-systems
- Design for performance and other system level considerations/constraints

Object Design

- Contains implementation details (data structures, algorithms, etc,)
- Focuses on “how” to do
- Follows the strategy established during system design

Implementation

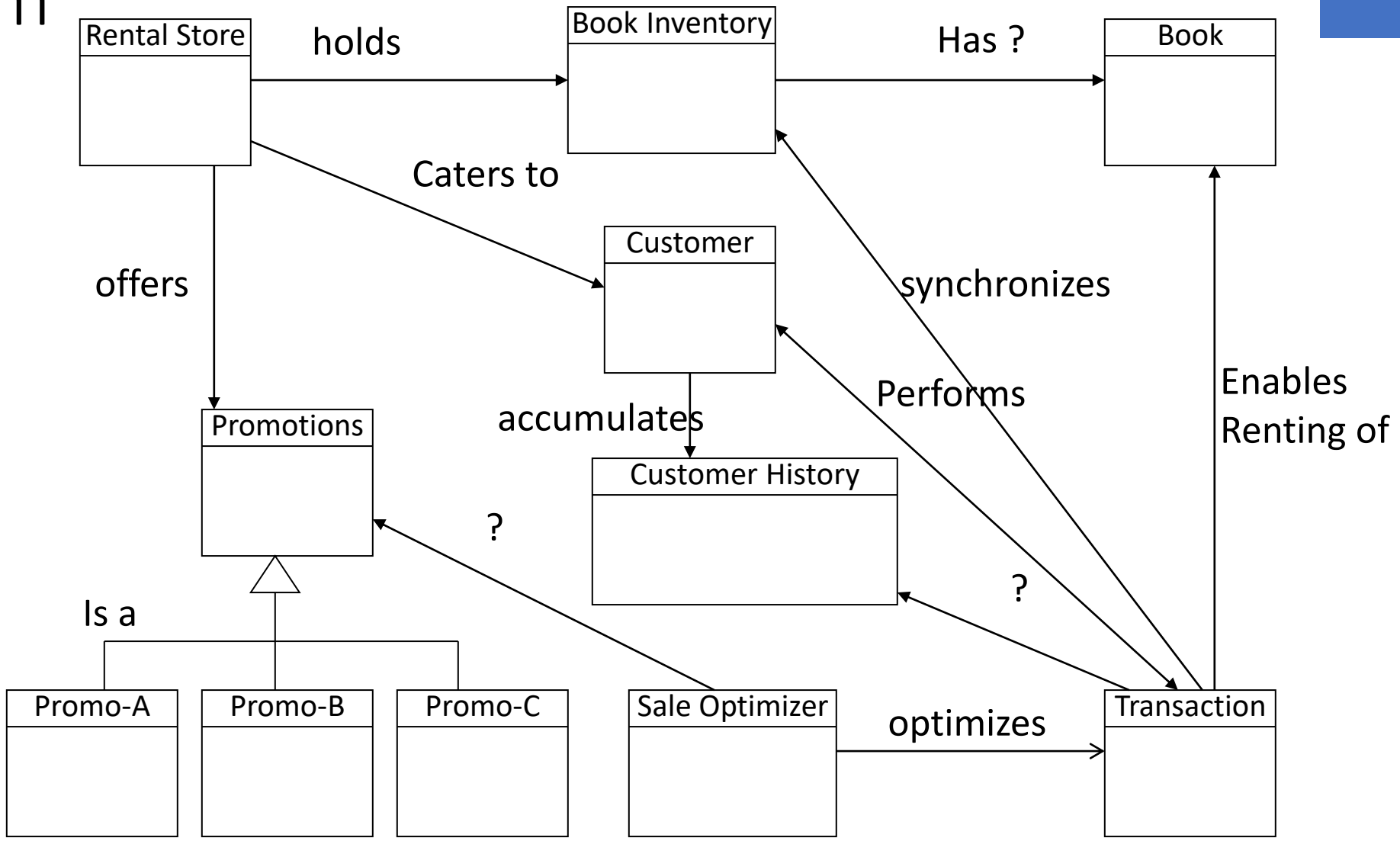
- Translation of Object Design into a programming language, database or hardware implementation
- All of the hard decisions should be made during design and implementation has to be relatively minor !
- Good software engineering practices are to be followed so that the implementation reflects the design

The 3 Models of OMT

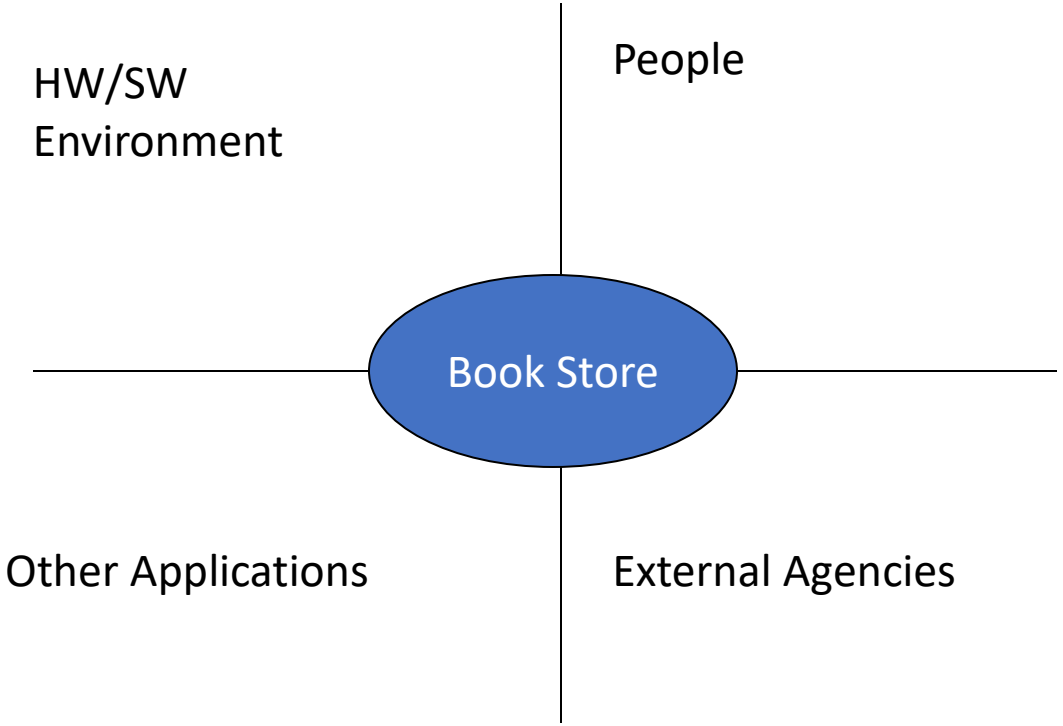
- Object Model (or Static Model) - contains object diagrams or class diagrams. Describes the static structure of the objects and their relationship
- Dynamic Model – Describes the dynamic behavior of the system and the interaction among objects. Uses state transition diagrams for example.
- Functional Model – Describes the data value transformation within a system. Contains data flow diagrams

  These 3 models are orthogonal parts of the description of complete system and are cross-linked.

Class Diagram



Book Store – Context Diagram



Book Store - Actors

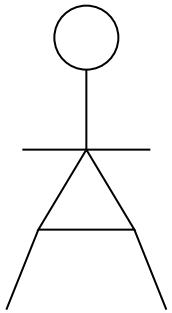
- Sales Person
- Manager
- CEO/Head
- Customer
- Visitor

Use Cases

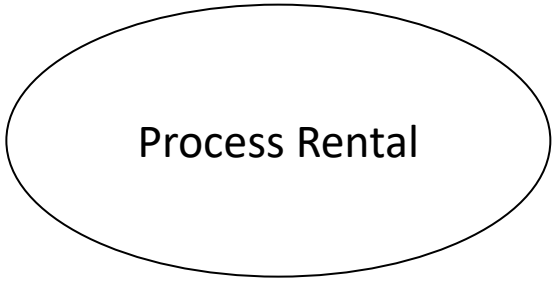
- Often ‘verbs’ in the problem statements reflect use cases. E.g.. Rent Books
- Each use case may have one or more actors associated
- Each use case may have multiple scenarios (each of them initiated by one and only actor)
- A use case can be an extension of an existing use case
- A use case may include an existing use case (Draw Cash includes Validate User)
- A use case may feature ‘generalization’ and some other use case(s) may inherit its behavior
- Organizing use cases help us identify Generalization, Include and Extend

Use Case Example

Sales Person

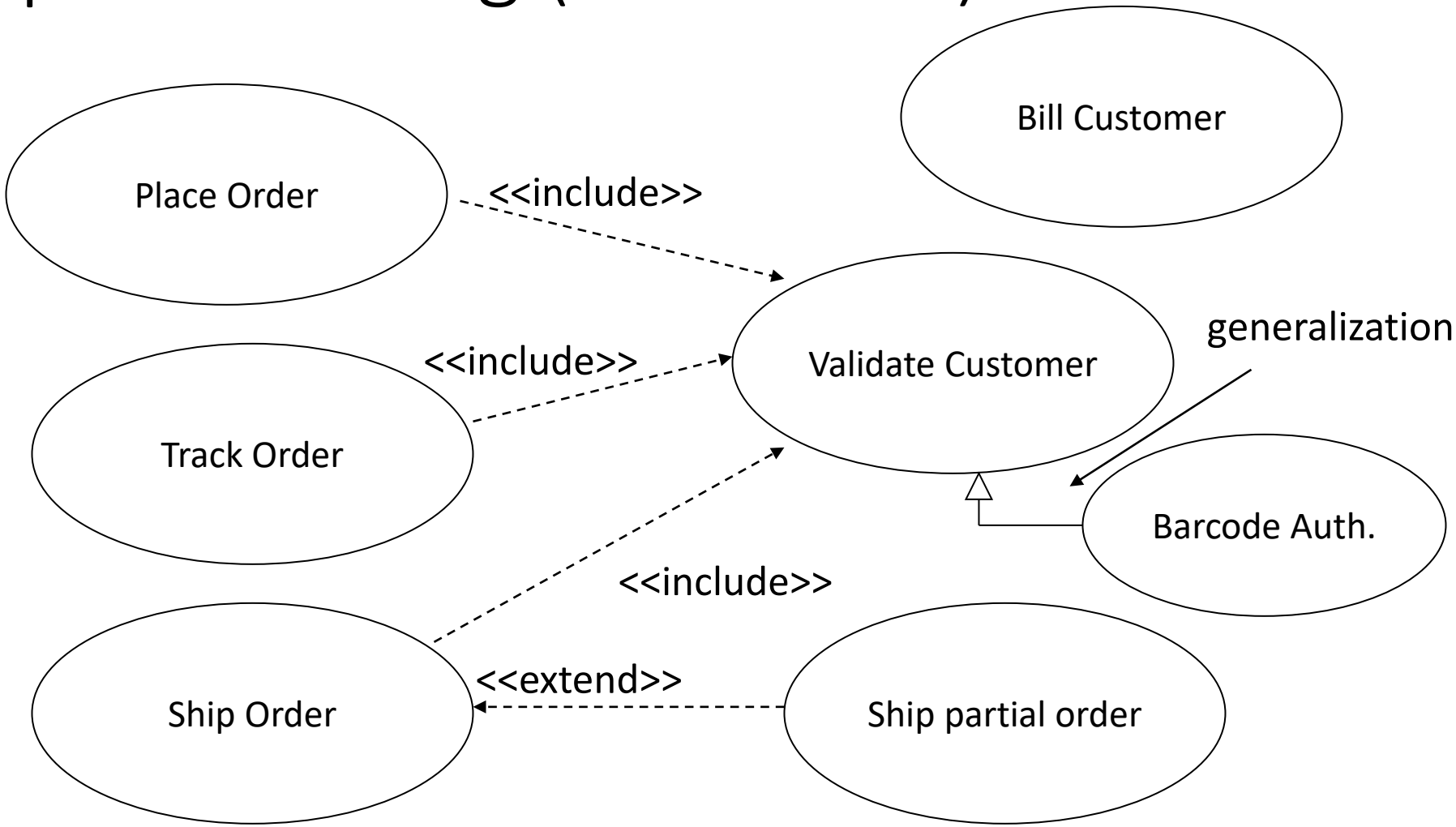


Actor

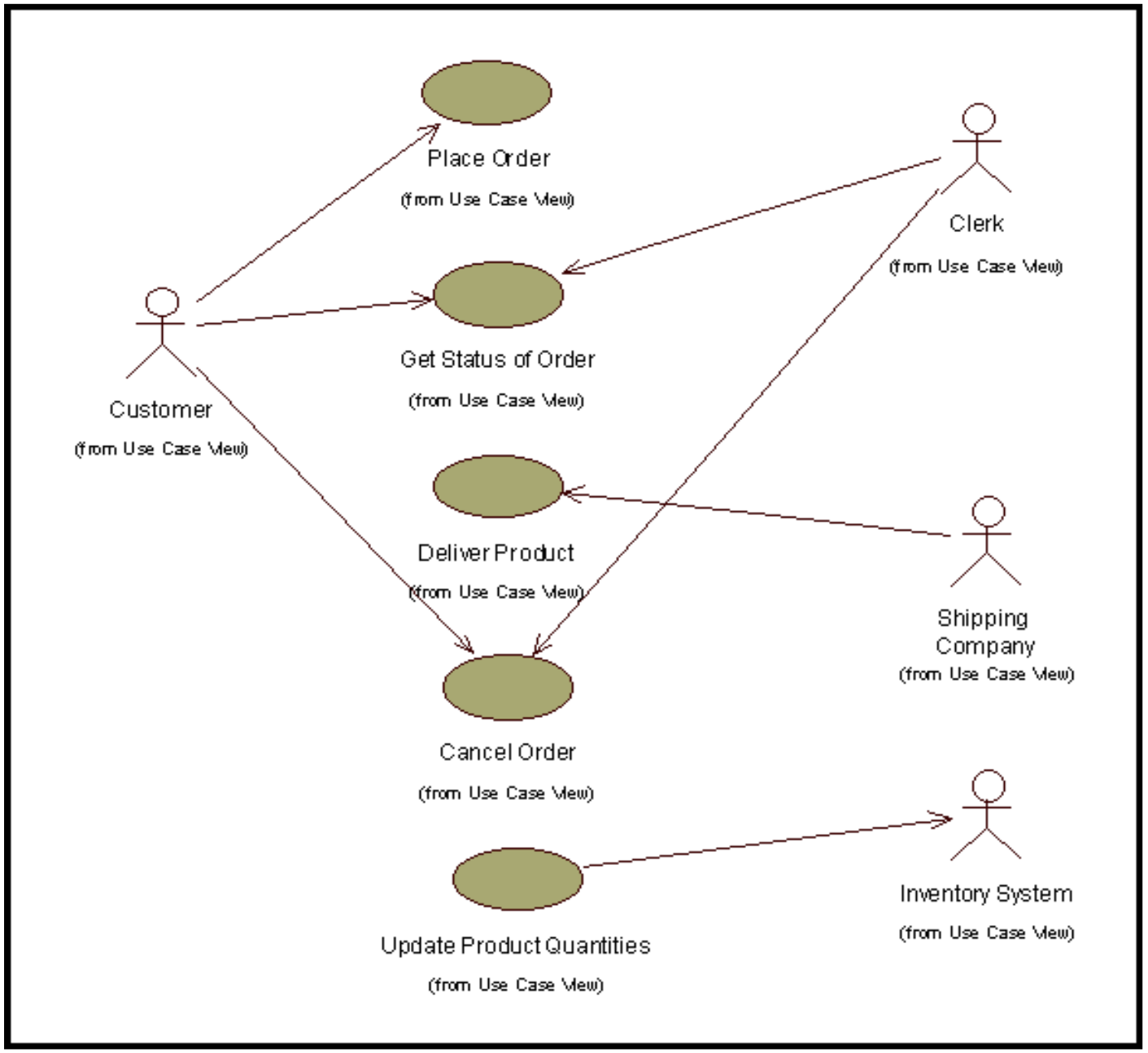


Use Case

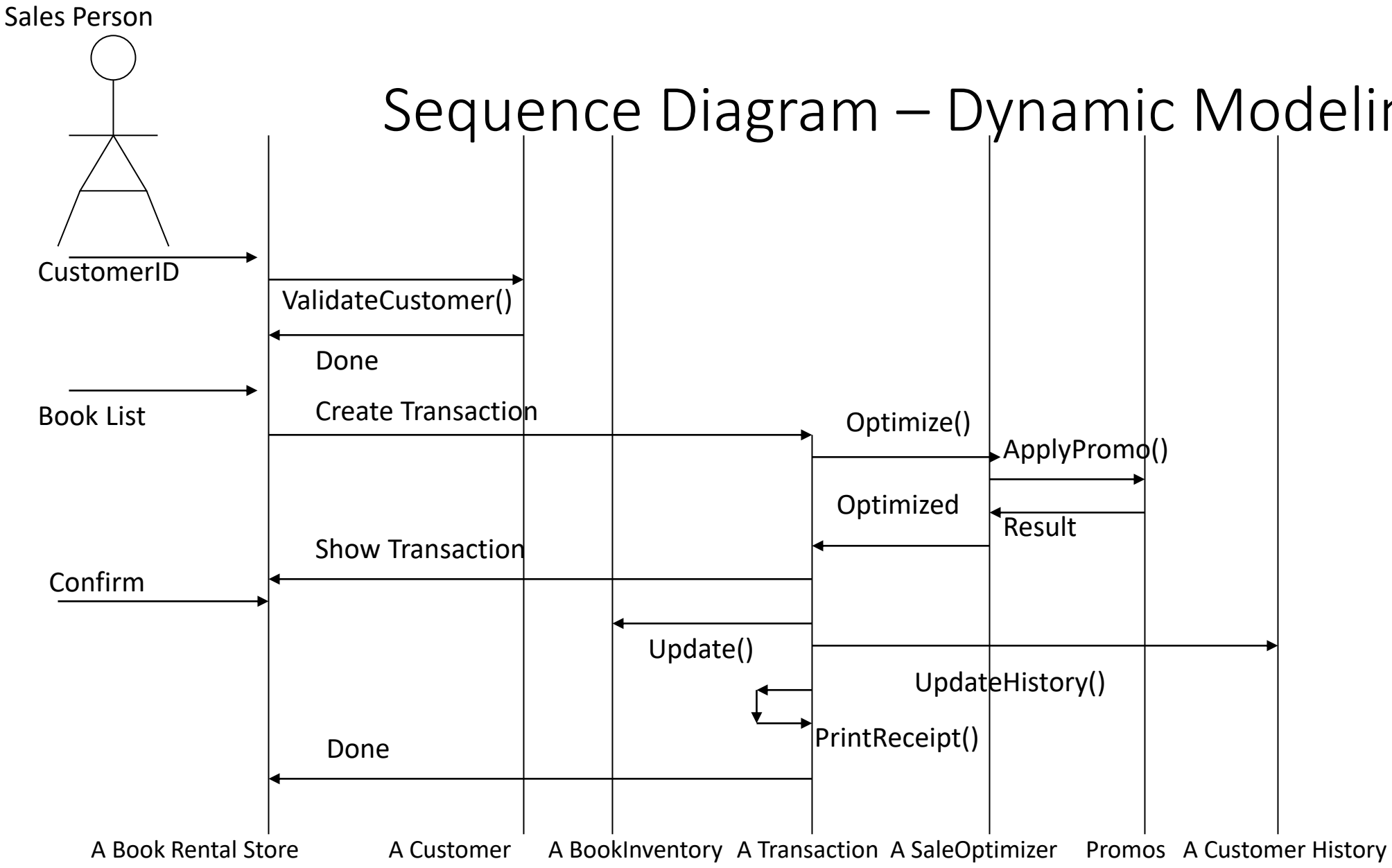
Sample Modeling (Use Cases)



Use Case Diagram



Sequence Diagram – Dynamic Modeling



CRC Cards (Class-Responsibility-Collaboration)

Name of the Class	
Responsibility-1	Collaborators
Responsibility-2	Collaborators
Responsibility-3	Collaborators
….	….
Responsibility-n	Collaborators

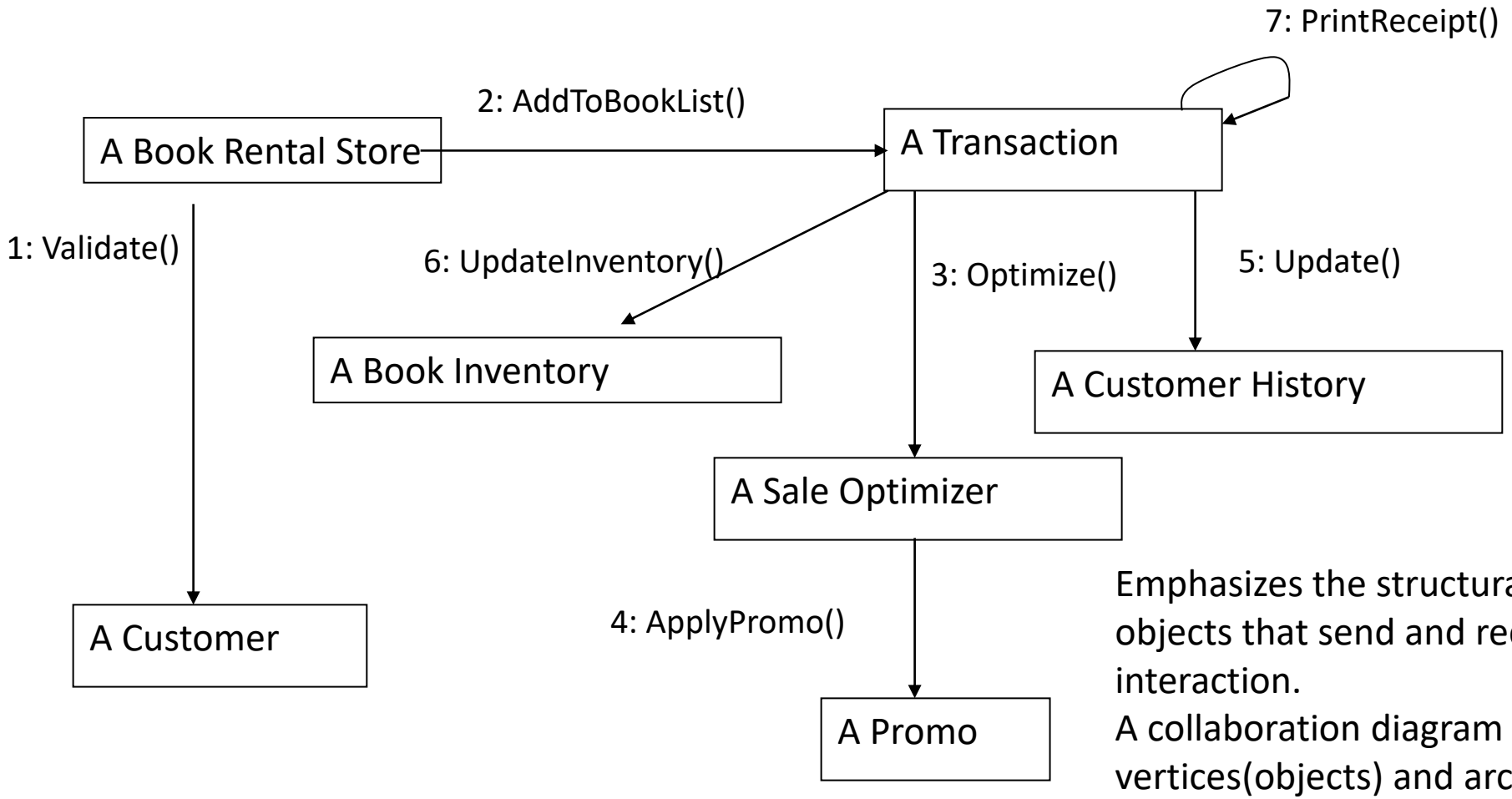
Gives us some idea about interaction/link between classes.

Class Diagrams and Interaction Diagrams (Sequence Diagrams and Collaboration Diagrams can be revised/reviewed using CRC cards.

CRC cards increase group interaction in teams.

It is good to write the list of responsibilities of each class in the Data Dictionary. It helps.

Collaboration Diagram

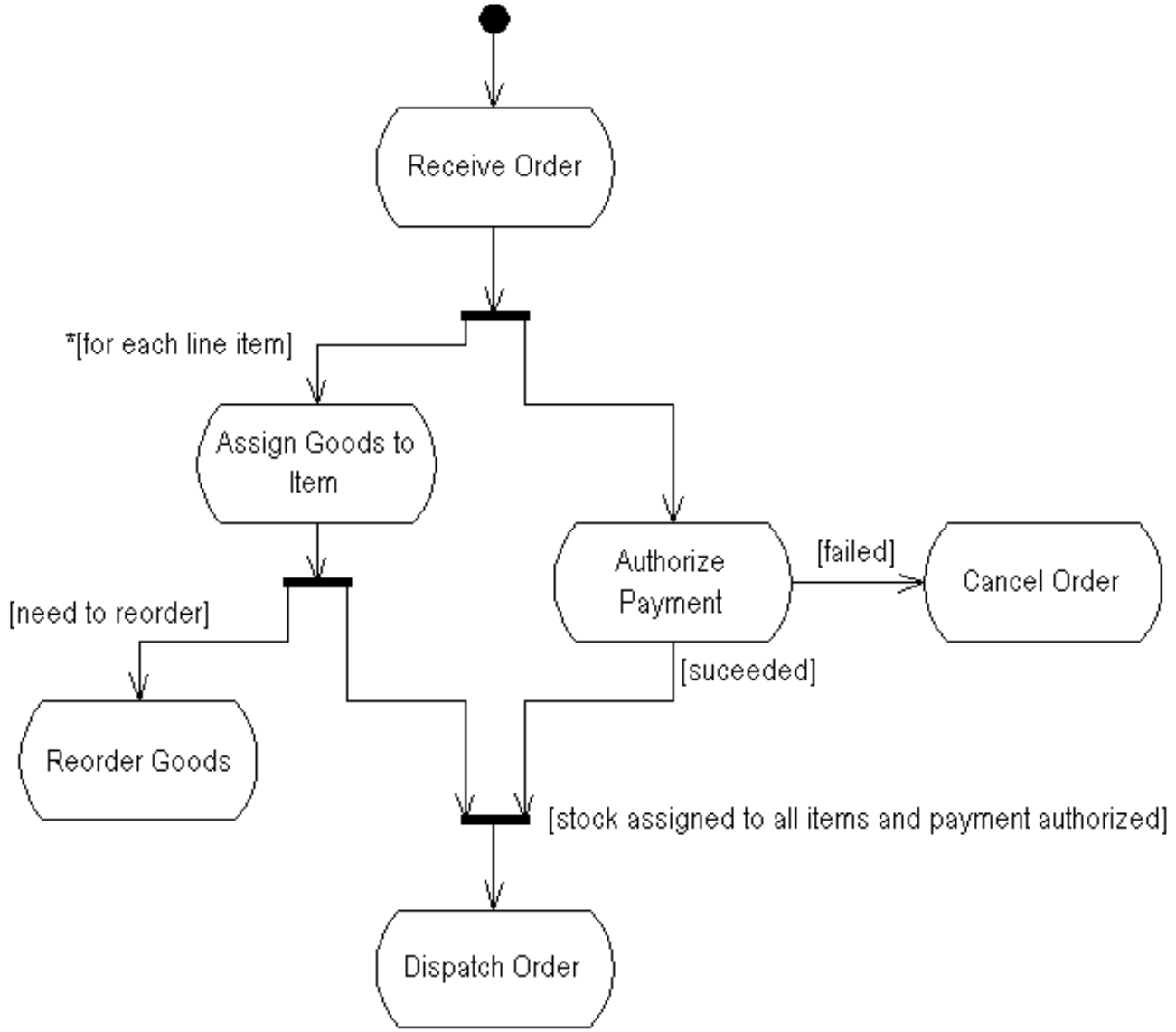


Emphasizes the structural organization of the objects that send and receive messages during an interaction.
A collaboration diagram is a collection of vertices(objects) and arcs (messages).

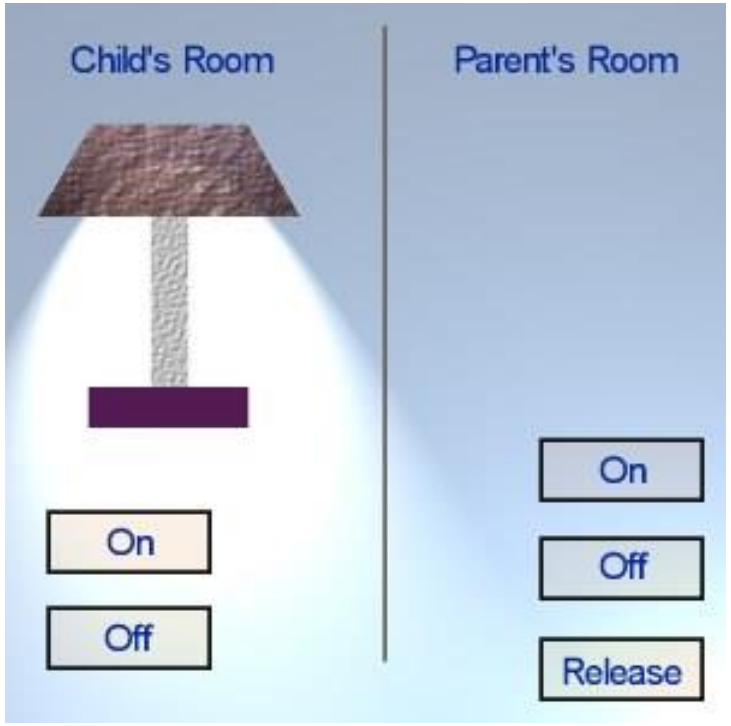
*There is no CheckInventory() ?

Activity Diagram

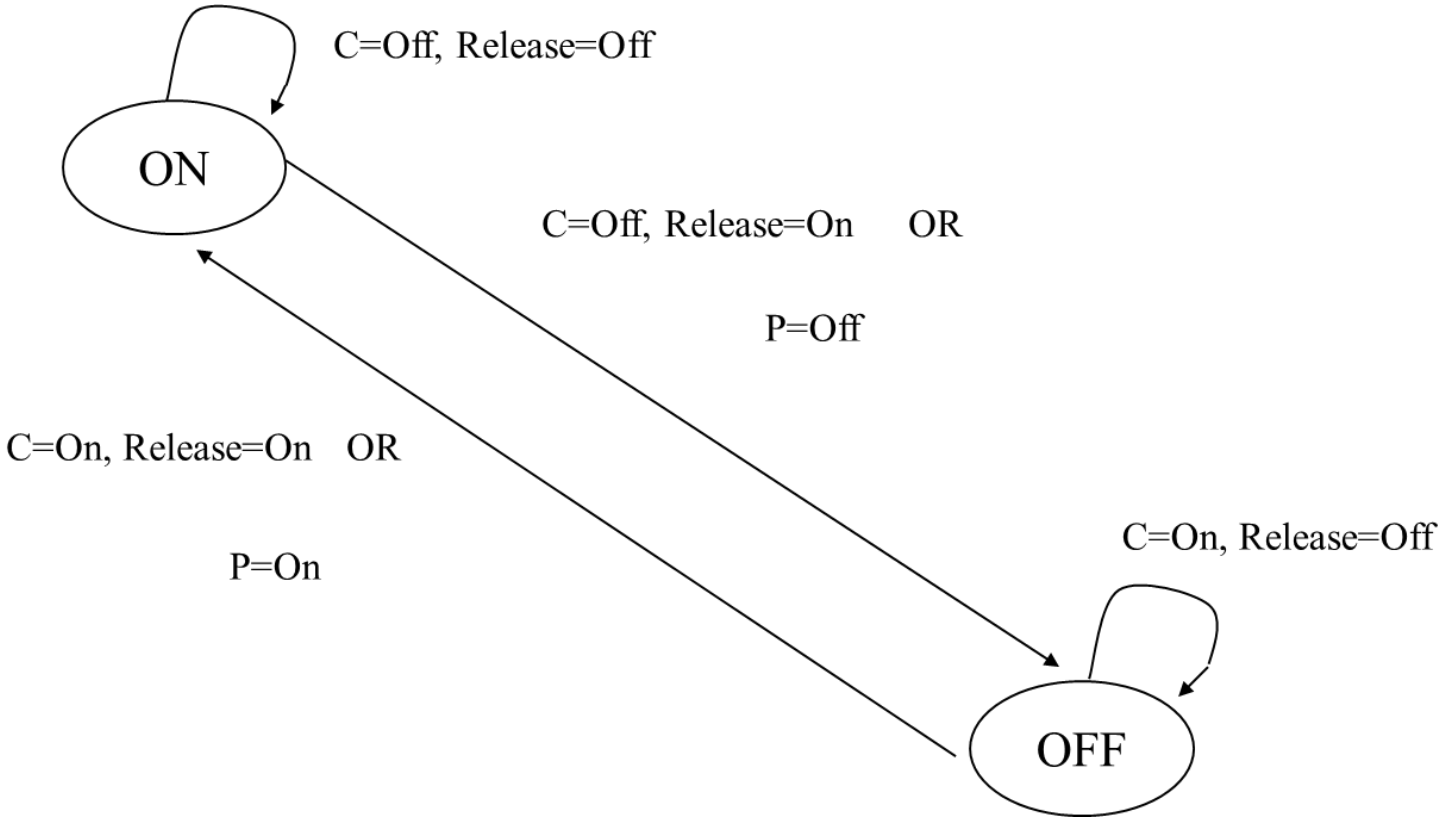
- Look similar to flow-chart at a high-level
- Show flow from activity to activity
- Used in dynamic modeling
- Used to understand the flow of control
- Help in forward and reverse engineering



State Transition Diagram (or State Chart Diagram)



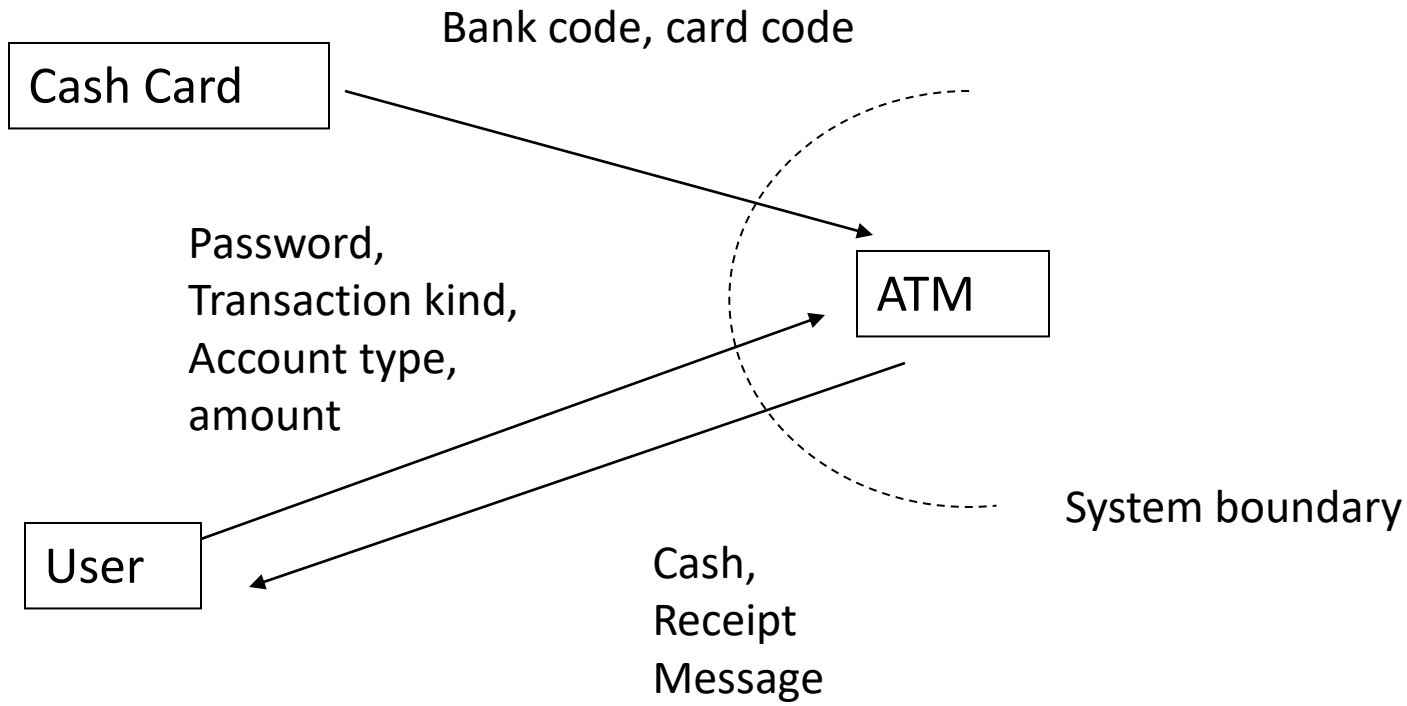
Design a state machine to encapsulate the behavior indicated above.



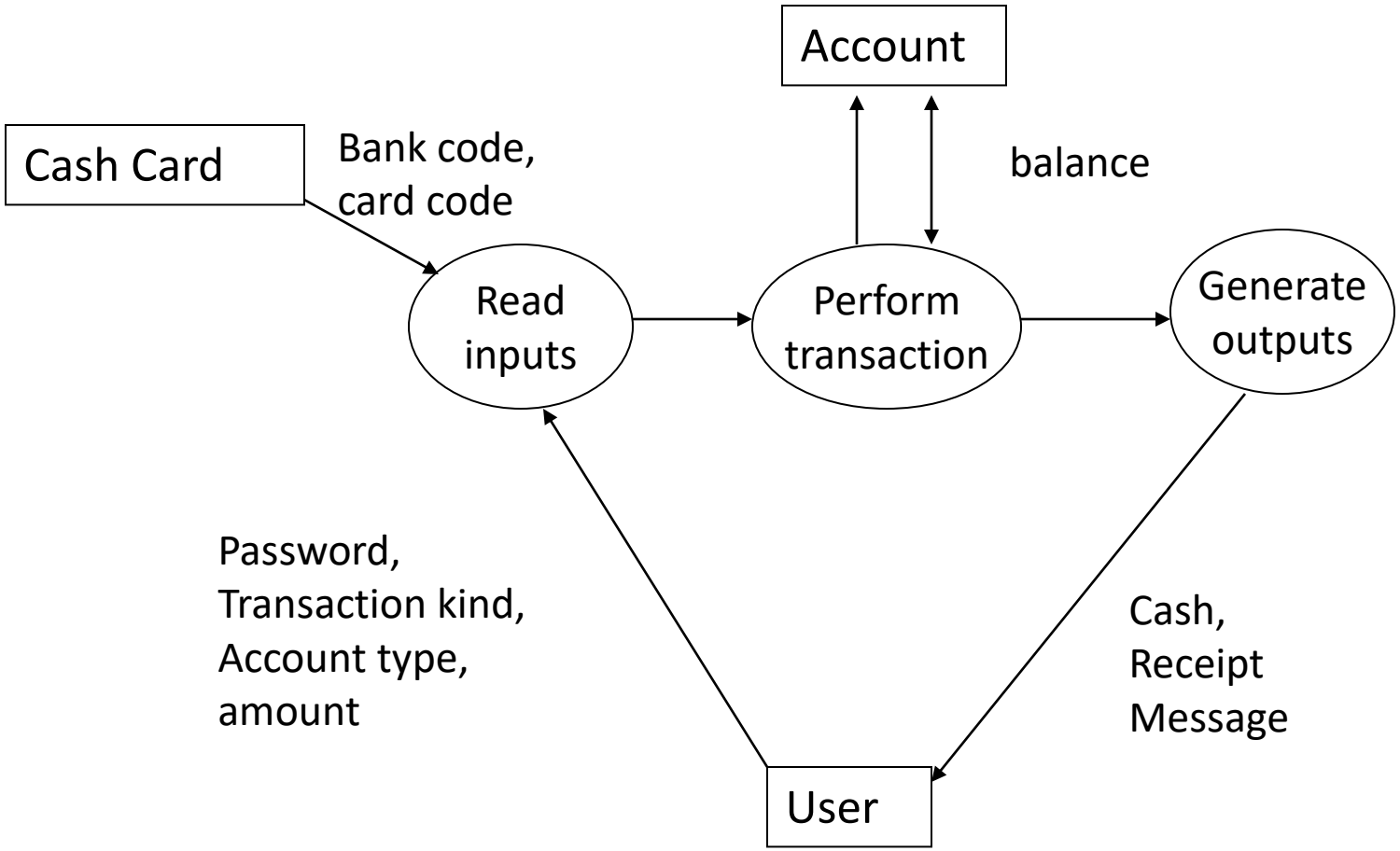
Analysis – Functional Modeling

- Identify input / output values
- Build data flow diagrams showing functional dependencies
- Describe functions, Identify constraints
- Specify optimization criteria

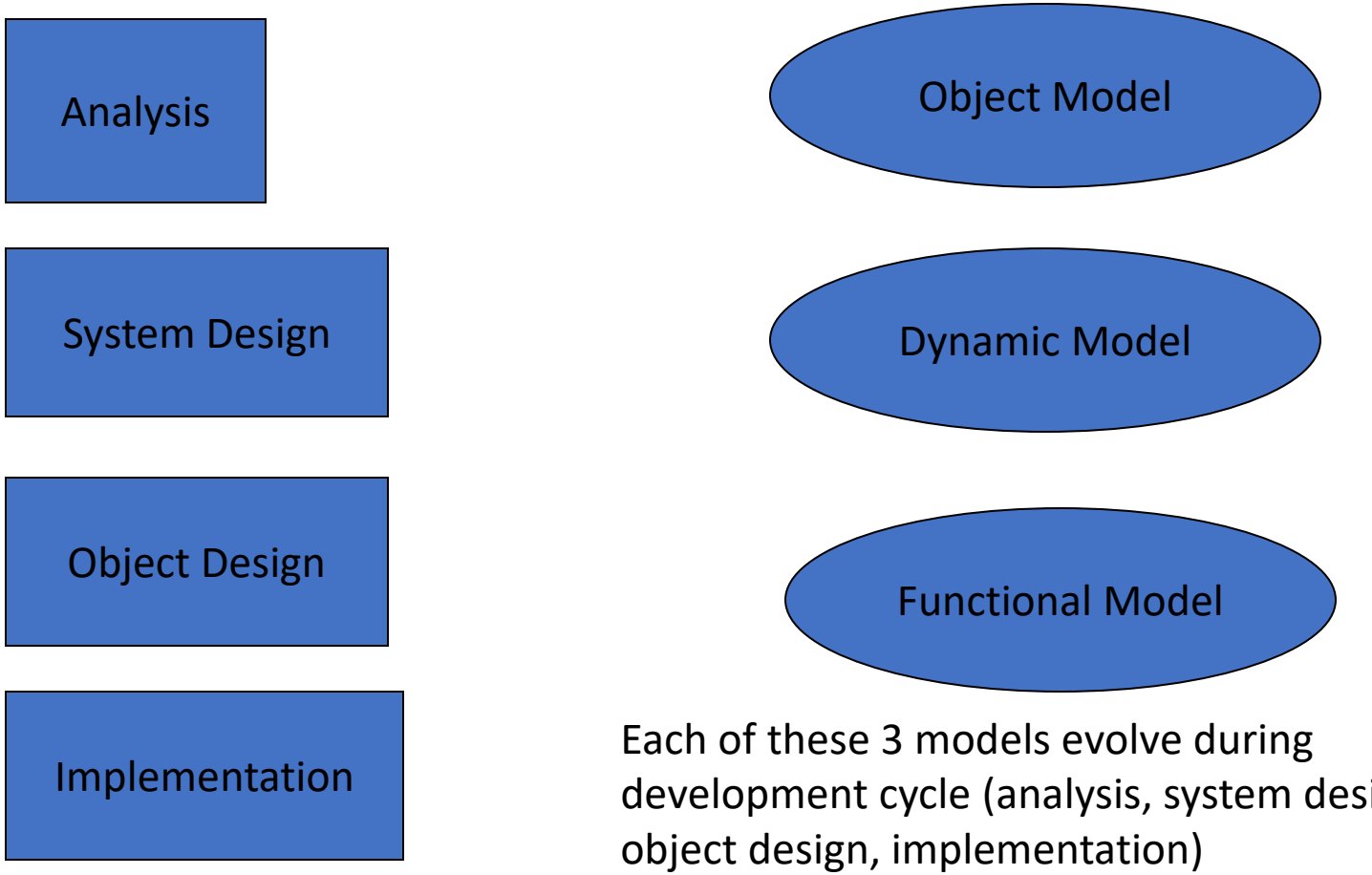
Input & Output values of ATM



Top level data flow diagram - ATM

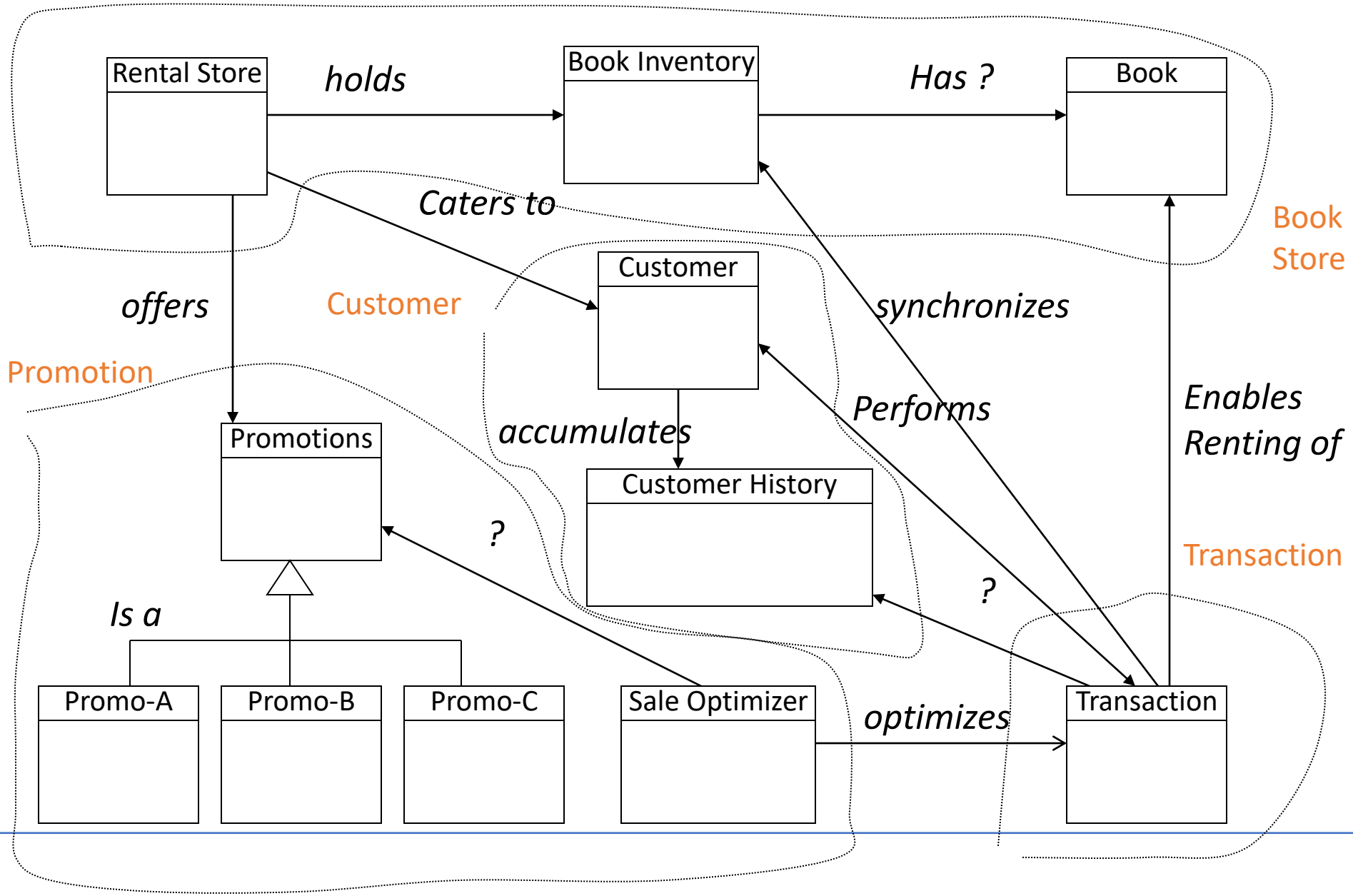


OMT – Models and Phases

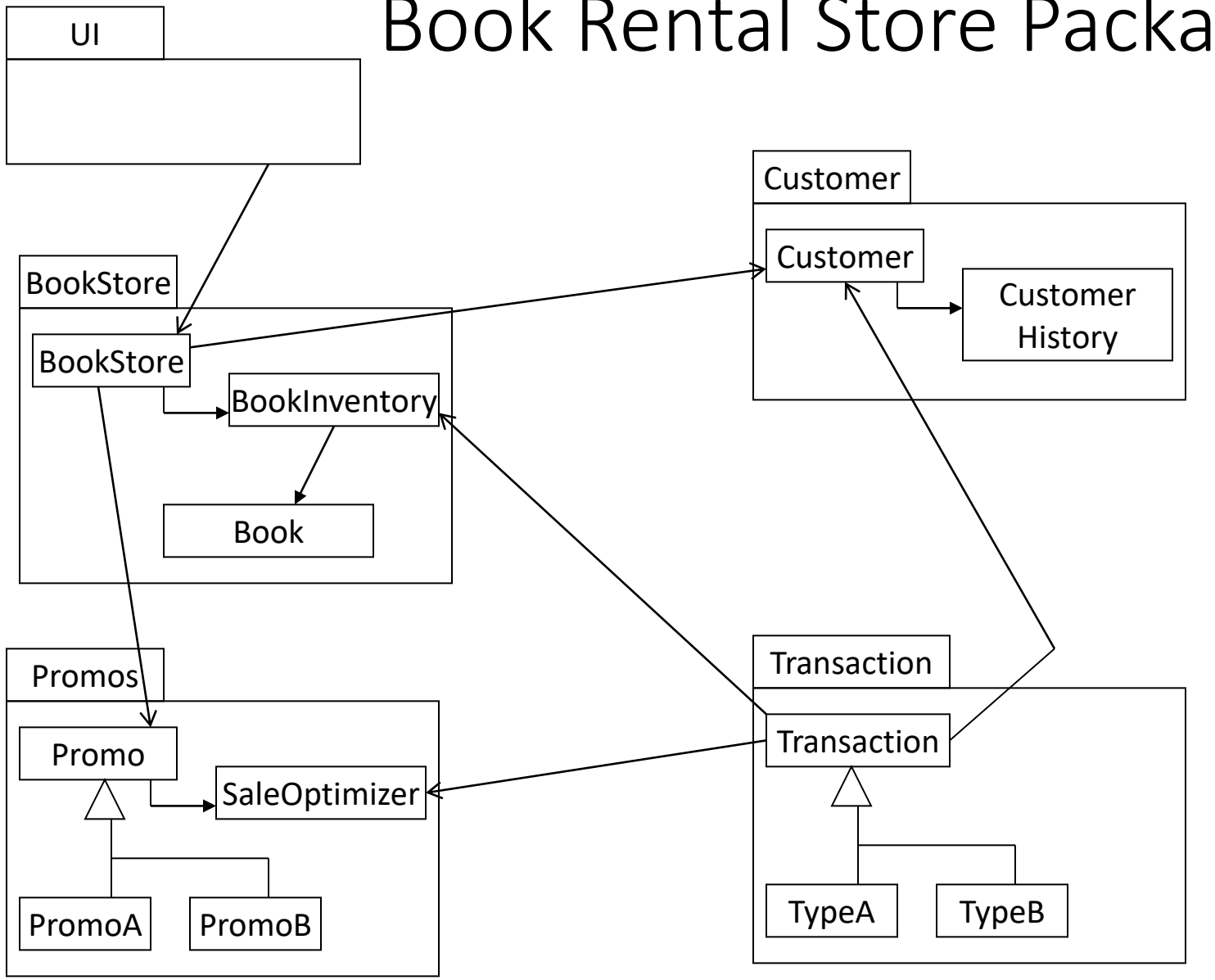


Each of these 3 models evolve during development cycle (analysis, system design, object design, implementation)

Book Rental Store Packages



Book Rental Store Package Diagram



UML – History

- Booch Method + OMT → Unified Method 0.8
- Unified Method 8.0 + OOSE (Jacobson) → UML 0.9
- UML 0.9 + other methods → UML 1.0
- Submission to OMG → UML 1.1
- Final submission to OMG and acceptance of UML 1.1
- Current version → UML 2.5.1

Summary

- ✓Introduction to Object Oriented Concepts
- ✓Class, Attributes, Methods and Object
- ✓Messages
- ✓Abstraction, Encapsulation and Data Hiding
- ✓Class Hierarchy, Inheritance and Polymorphism
- ✓Relationships (Is-a, Has-a, Uses-a)
- ✓Role of UML in OOAD
- ✓Object Modeling Technique

Question 6

What are the three models of Object Modeling Technique (OMT)?

A. Object Model (or Static Model), Dynamic Model, and Functional Model
B. Object Model, Behavior Model, Functional Model
C. Object Model, Sequence Model, Functional Model
D. Object Model, Dynamic Model, Technology Model

ANSWER: A

Question 7

Context diagram helps us

A. Understand use cases and exceptional flows
B. Understand different states of the context
C. Model the context and understand possible actors
D. Identify the non-functional requirements

ANSWER: C

Question 8

In a sequence diagram a vertical line represents

A. Sequence of actions
B. An object instantiated from a class
C. Start and finish of a sequence
D. Method invocation or a message

ANSWER: B

Question 9

In a sequence diagram a horizontal line or arrow from left to right represents

A. Sequence of actions
B. An object instantiated from a class
C. Start and finish of a sequence
D. Method invocation or a message

ANSWER: D

Question 10

Which phase of OMT focuses on writing object-oriented source code?

A. Object Design
B. Implementation
C. System Design
D. Source Code Writing

ANSWER: B

Thank You!