Software Design

Object Oriented Analysis and Design

A Design Process

Developed from various methodologies.

-However, UML has been designed to be usable with any design methodology

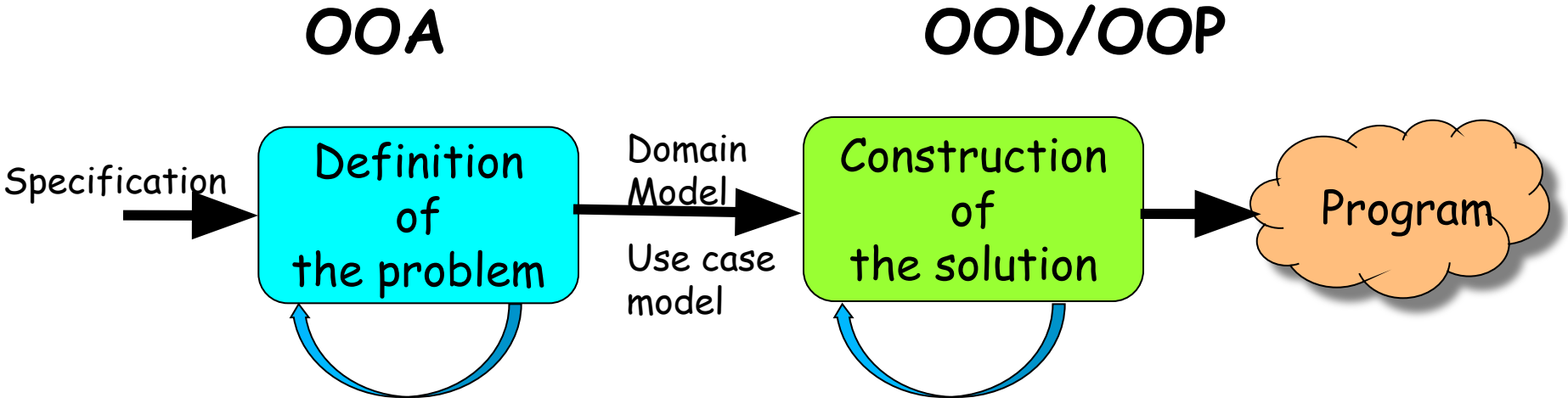
From requirements specification, initial model is developed (OOA)

-Analysis model is iteratively refined into a design model

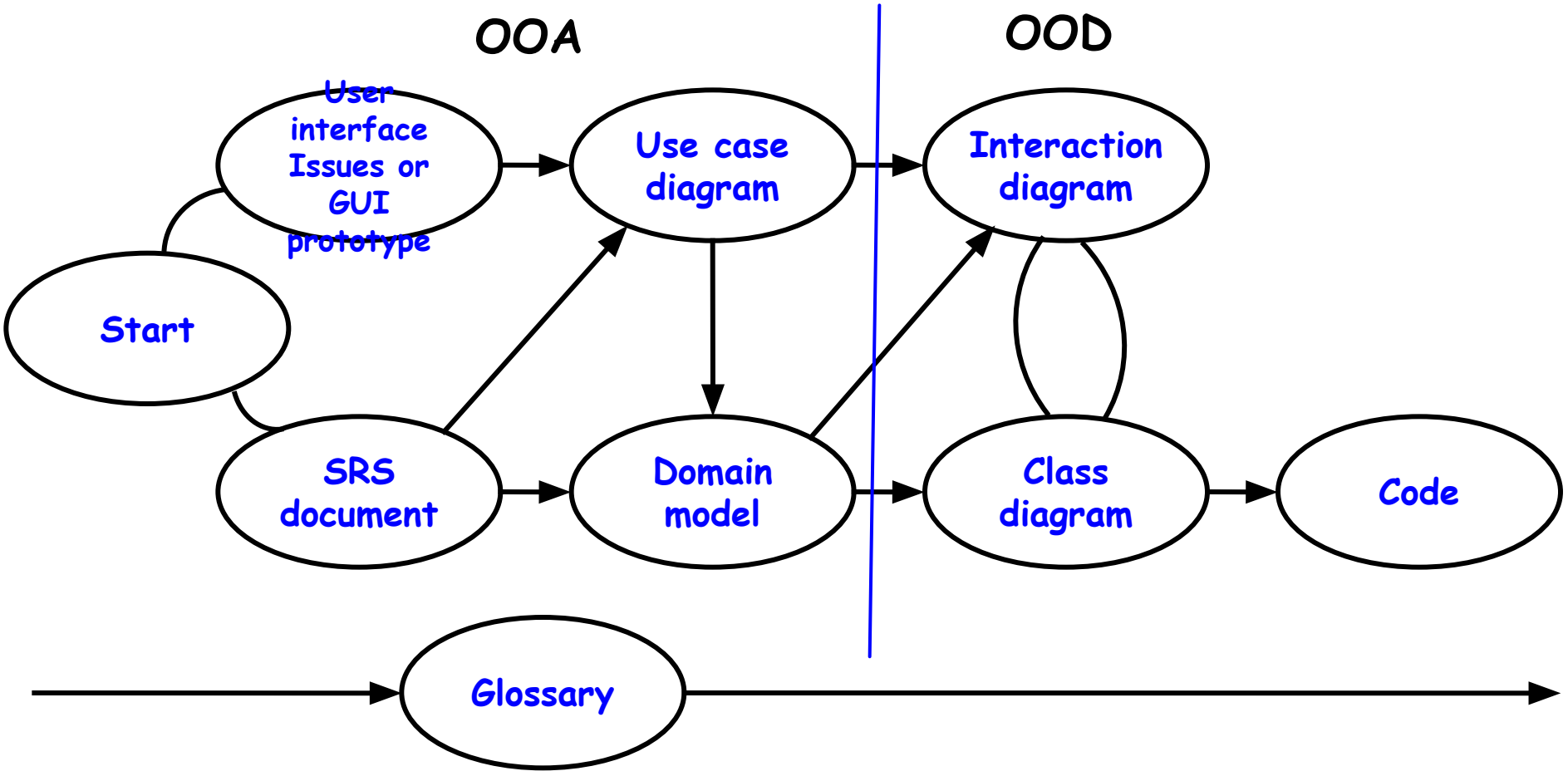
Design model is implemented using OO concepts

OOAD

Iterative and Incremental



Unified Development Process Cont…



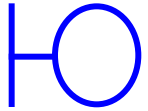
Domain Modelling

- Represents concepts or objects appearing in the problem domain.
- Also captures relationships among objects.
- Three types of objects are identified
  - Boundary objects
  - Entity objects
  - Controller objects

Class Stereotypes

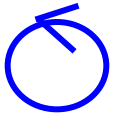
Three different stereotypes on classes are used: <<boundary>>, <<control>>, <<entity>>.

Boundary



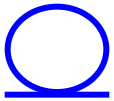
Cashier Interface

Control



Withdrawal

Entity



Account

Boundary Objects

- Interact with actors:
  - User interface objects
- Include screens, menus, forms, dialogs etc.
- Do not perform processing but validates, formats etc.

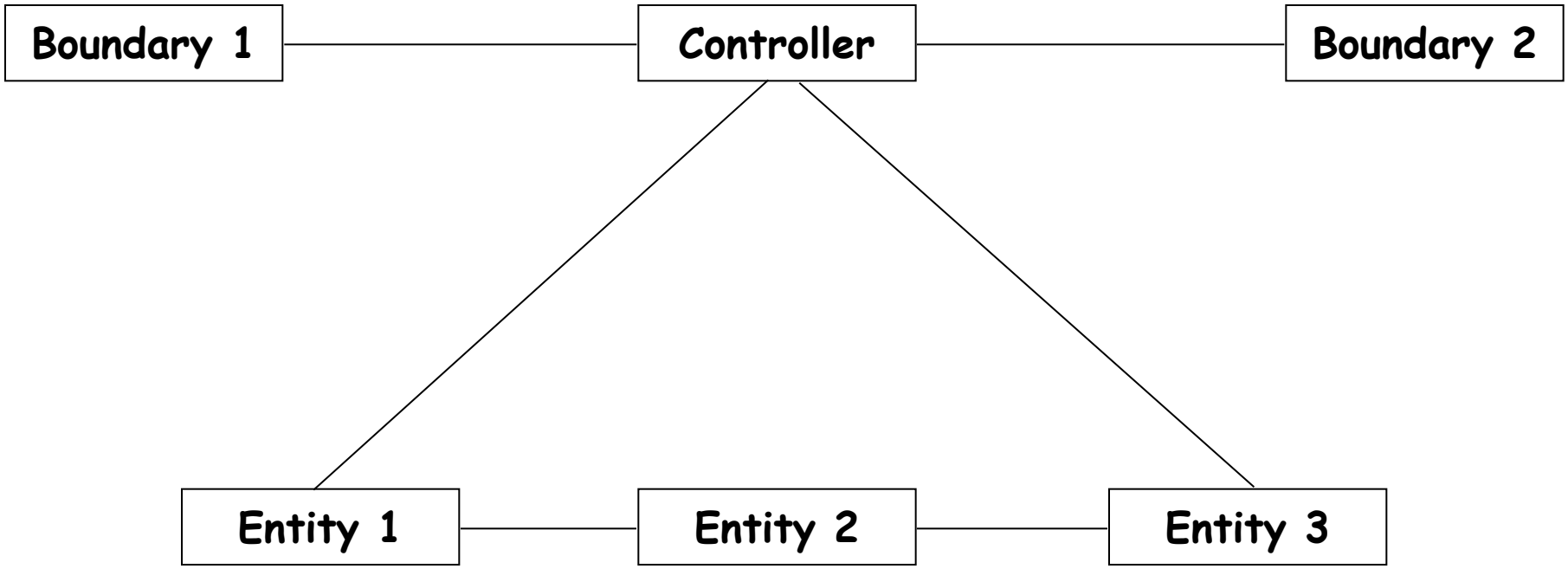
Entity Objects

- Hold information:
  - Such as data tables & files, e.g. Book, BookRegister
- Normally are dumb servers
- Responsible for storing data, fetching data etc.
- Elementary operations on data such as searching, sorting, etc.
- Entity Objects are identified by examining nouns in problem description

Controller Objects

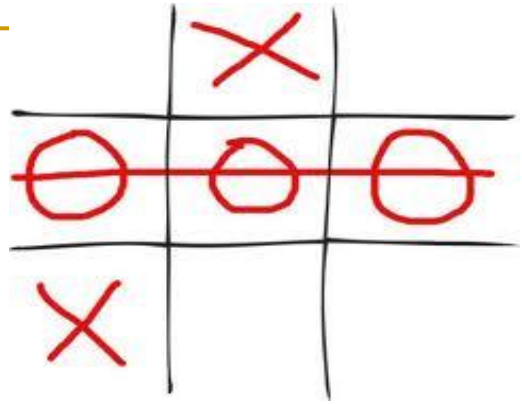
- Coordinate the activities of a set of entity objects
- Interface with the boundary objects
- Realizes use case behavior
- Embody most of the logic involved with the use case realization
- There can be more than one controller to realize a single use case

Use Case Realization



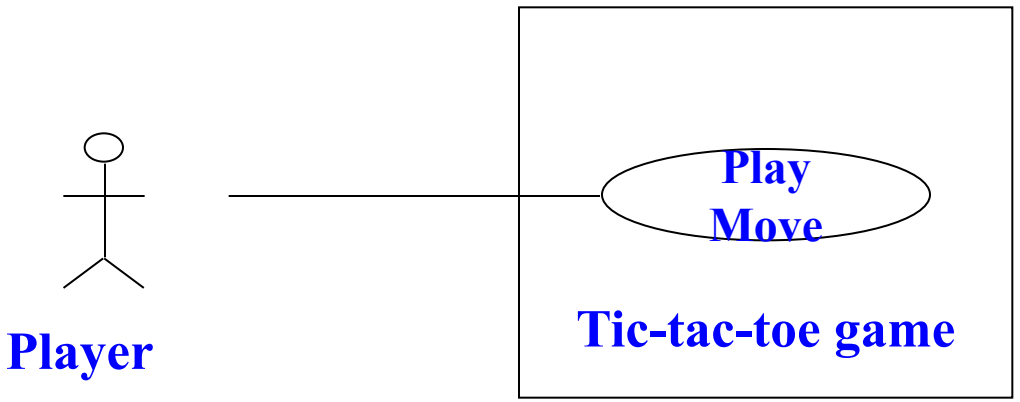
Realization of use case through the collaboration of Boundary, controller and entity objects

Example 1 : Tic-tac-toe



- Tic-tac-toe is a computer game in which a human player and the computer make alternative moves on a 3×3 square.
- A **move** consists of marking previously unmarked square.
- The **player plays** by placing three consecutive marks along a straight line on the square (i.e. along a row, column, or diagonal) wins the game.
- As soon as either the human player or the computer wins, a message congratulating the winner should be displayed.
- If neither player manages to get three consecutive marks along a straight line, but all the squares on the board are filled up, then the game is drawn.
- The computer always tries to win a game.

Example 1: Use Case Model

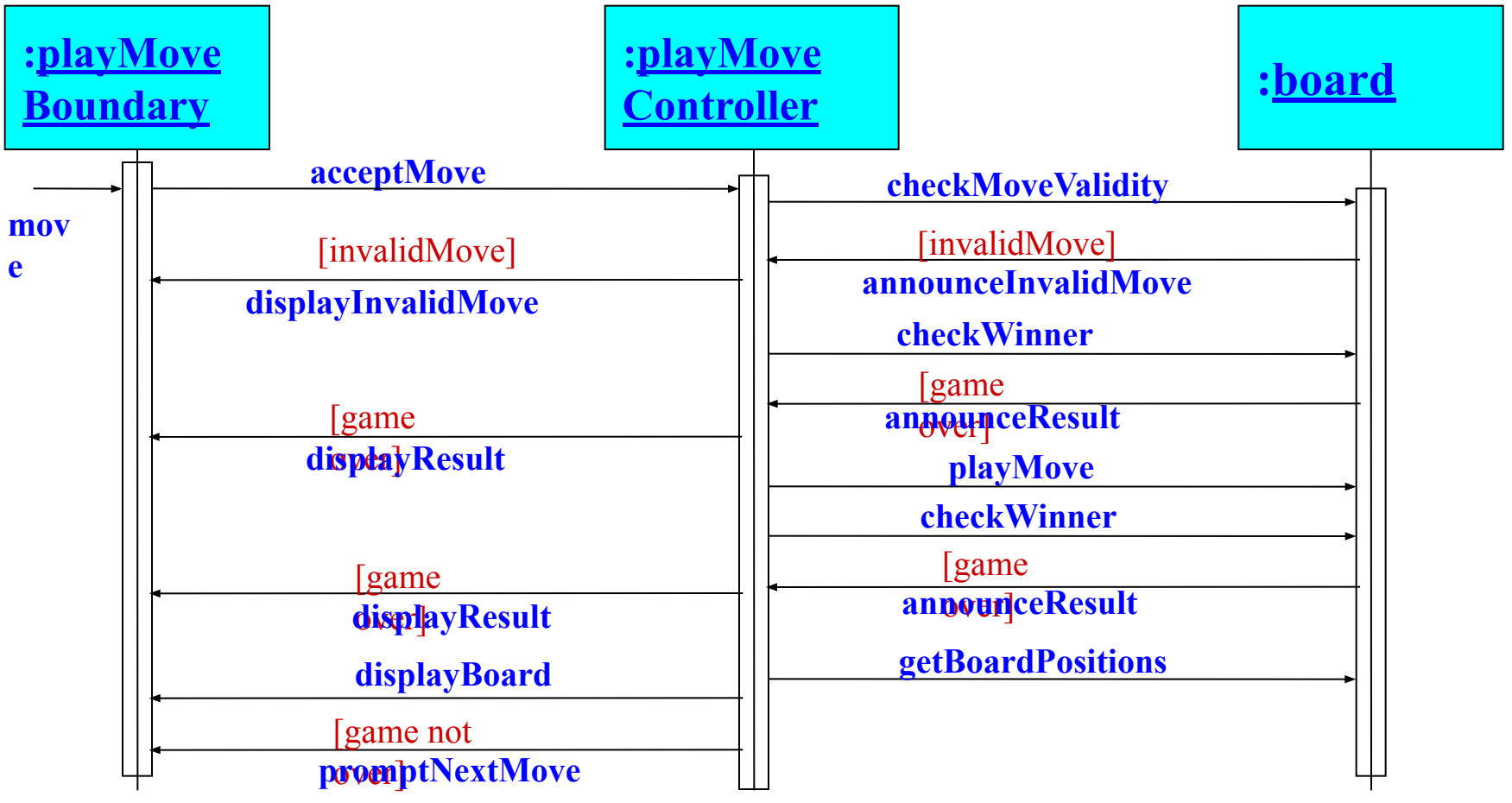


Example 1: Initial and Refined Domain Model

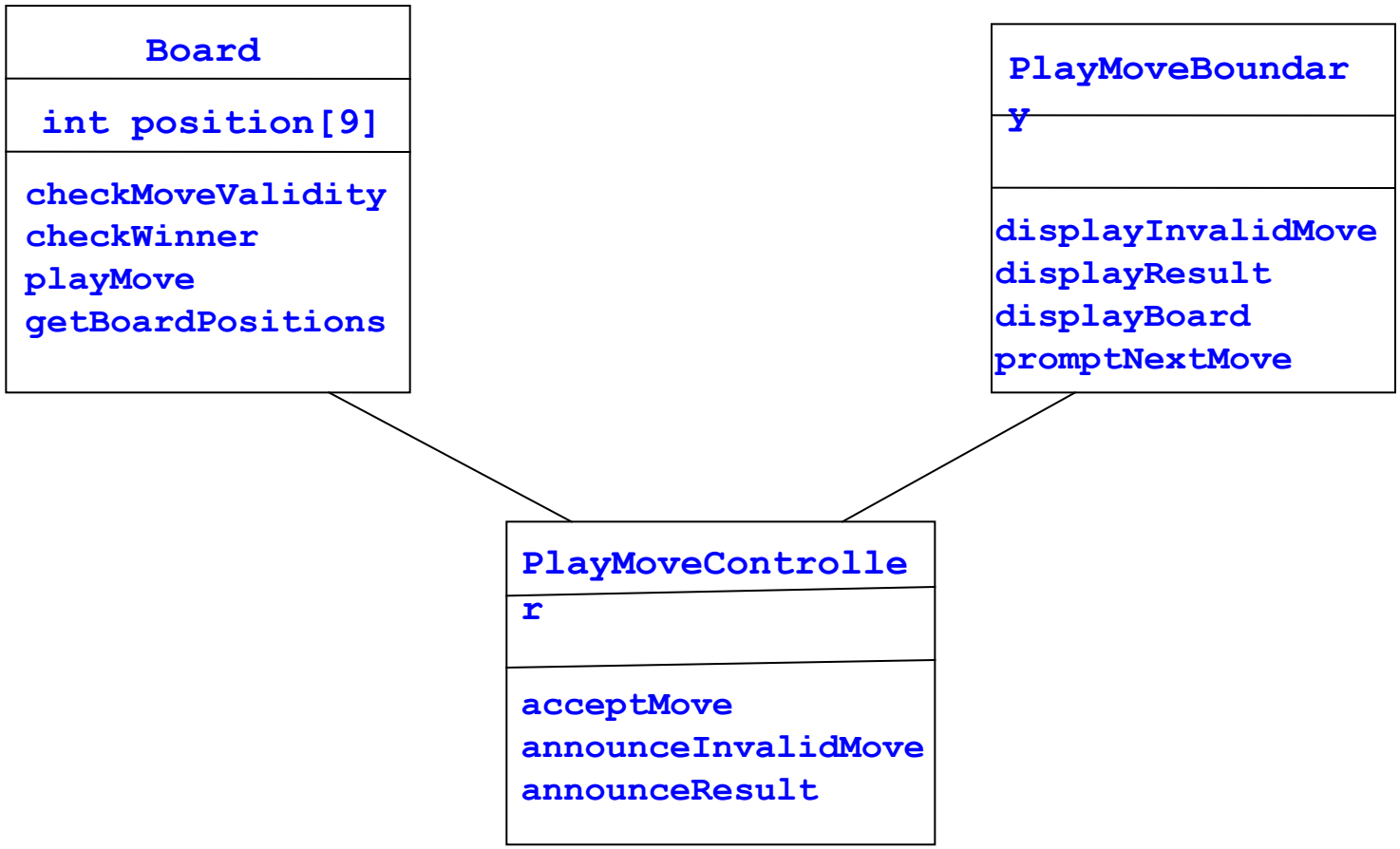
Initial domain model

Refined domain model

Example 1: Sequence Diagram



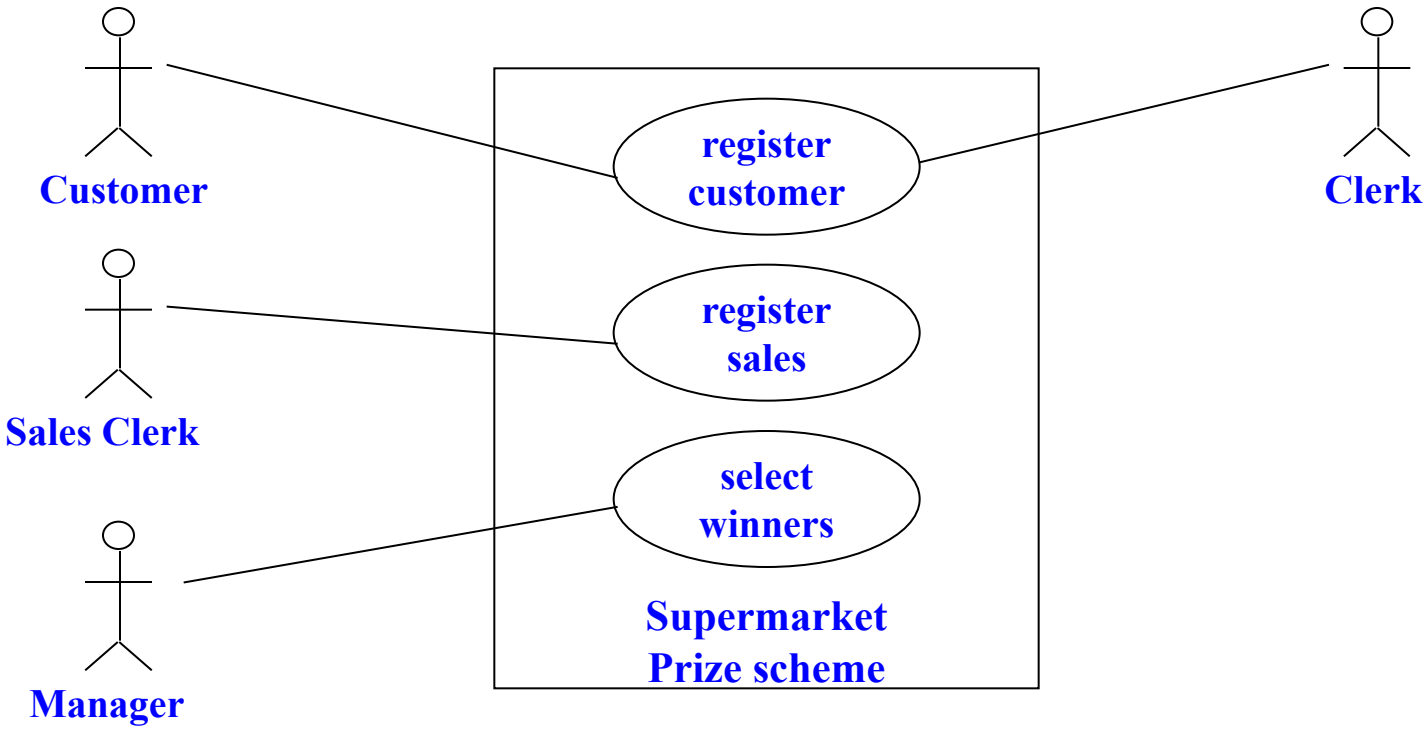
Example 1: Class Diagram



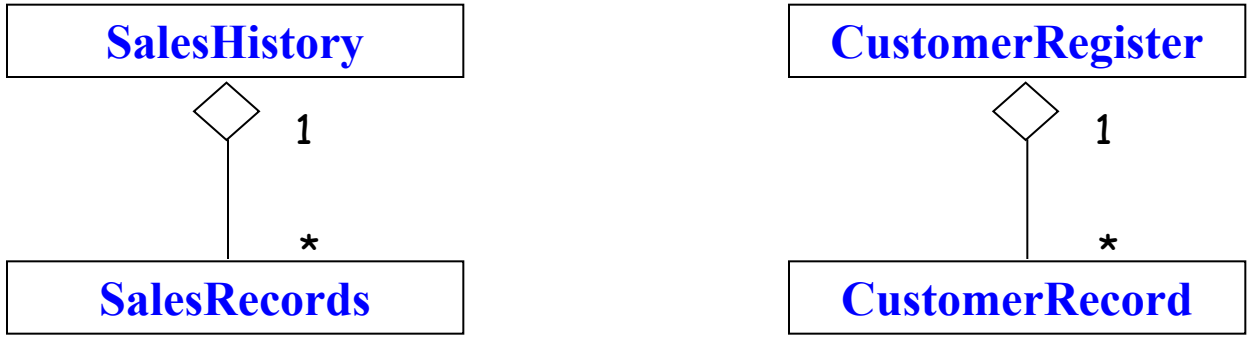
Example 2 : Supermarket Prize Scheme

- A supermarket needs to develop the following software to encourage regular customers.
- For this, the customer needs to supply his/her residence address, telephone number, and the driving license number. Each customer who registers for this scheme is assigned a unique customer number (CN) by the computer.
- A customer can present his CN to the check out staff when he makes any purchase.
- In this case, the value of his purchase is credited against his CN.
- At the end of each year, the supermarket intends to award surprise gifts to 10 customers who make the highest total purchase over the year.
- Also, it intends to award a gold coin to every customer whose purchase exceeded Rs.100,000.
- The entries against the CN are the reset on the day of every year after the prize winners' lists are generated.

Example 2: Use Case Model

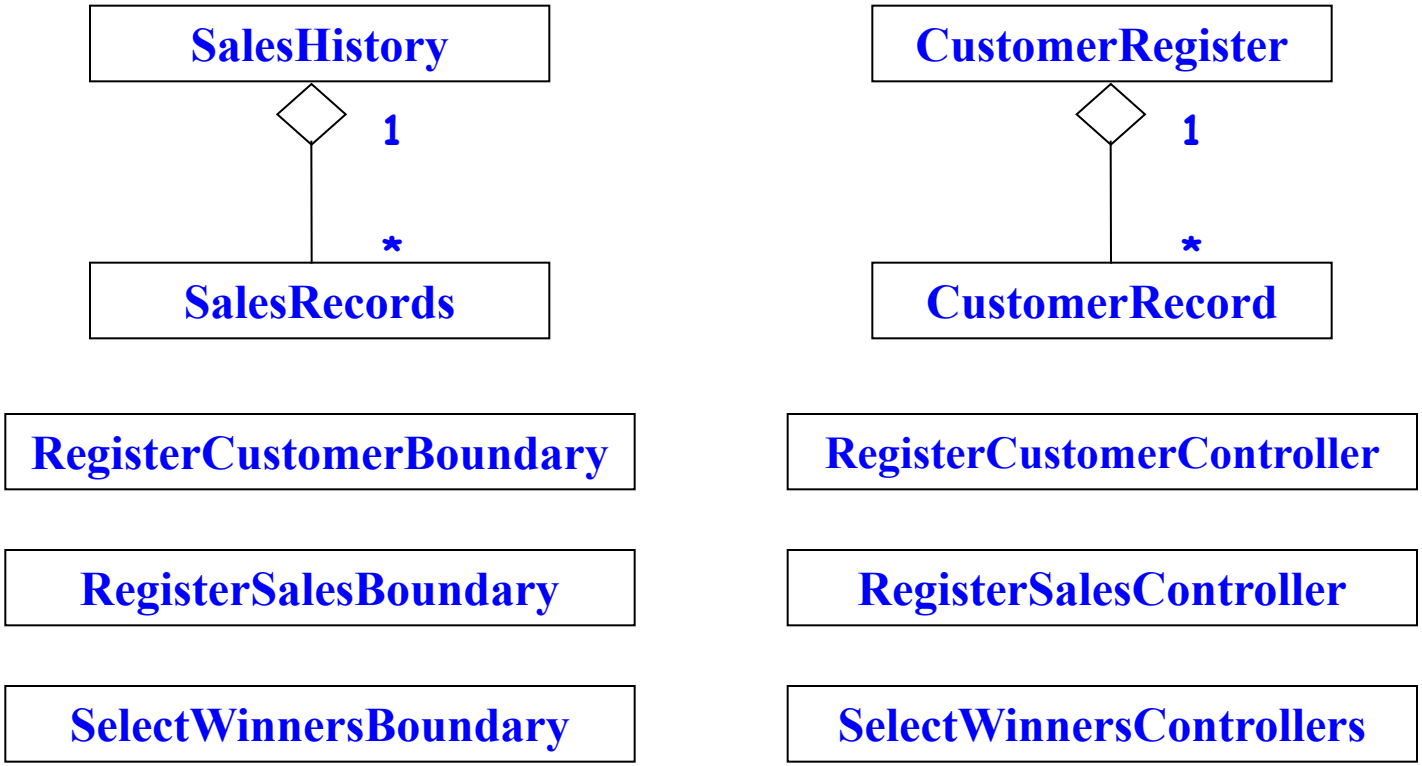


Example 2: Initial Domain Model



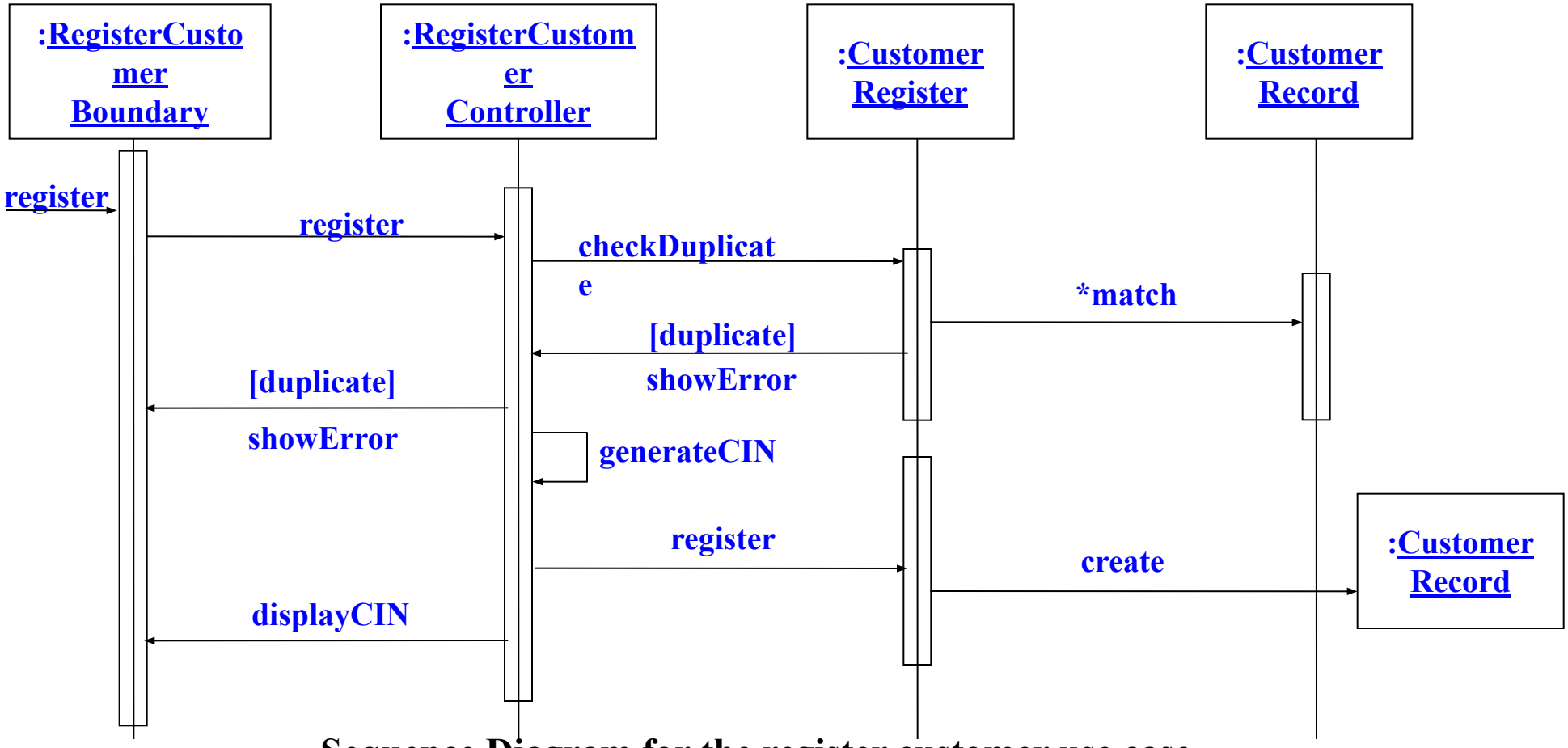
Initial domain model

Example 2: Refined Domain Model



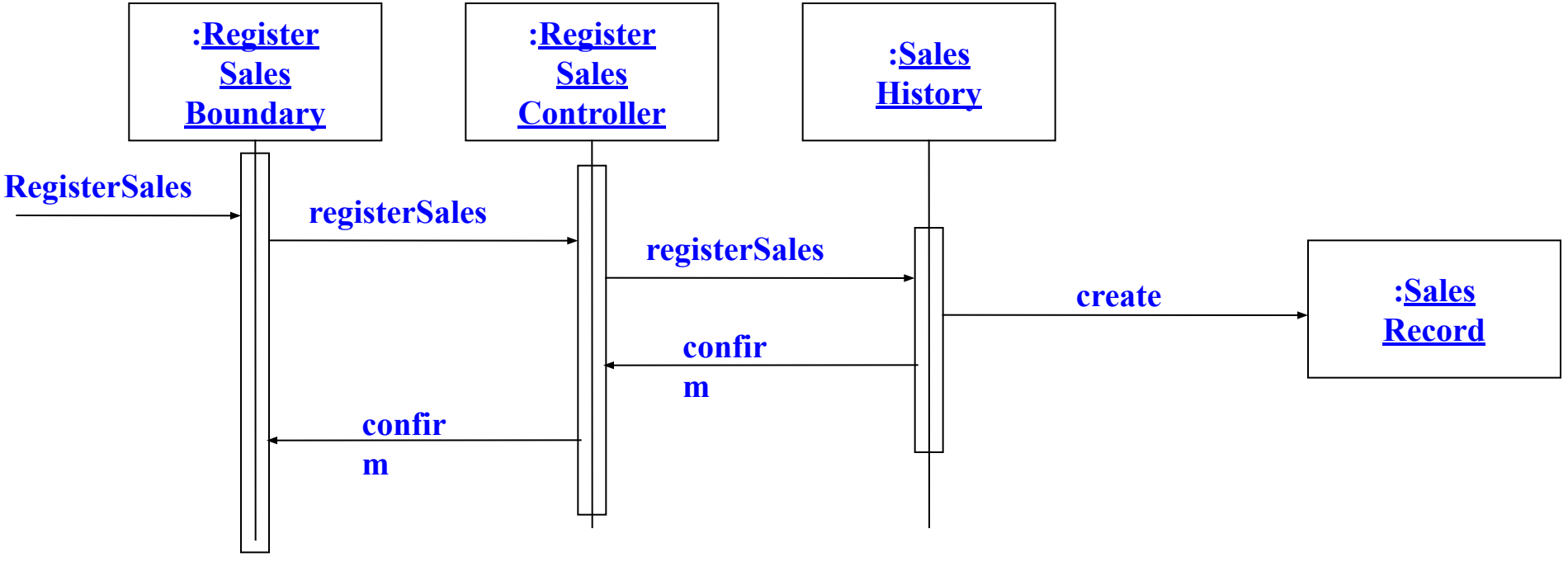
Refined domain model

Example 2: Sequence Diagram for the Register Customer Use Case



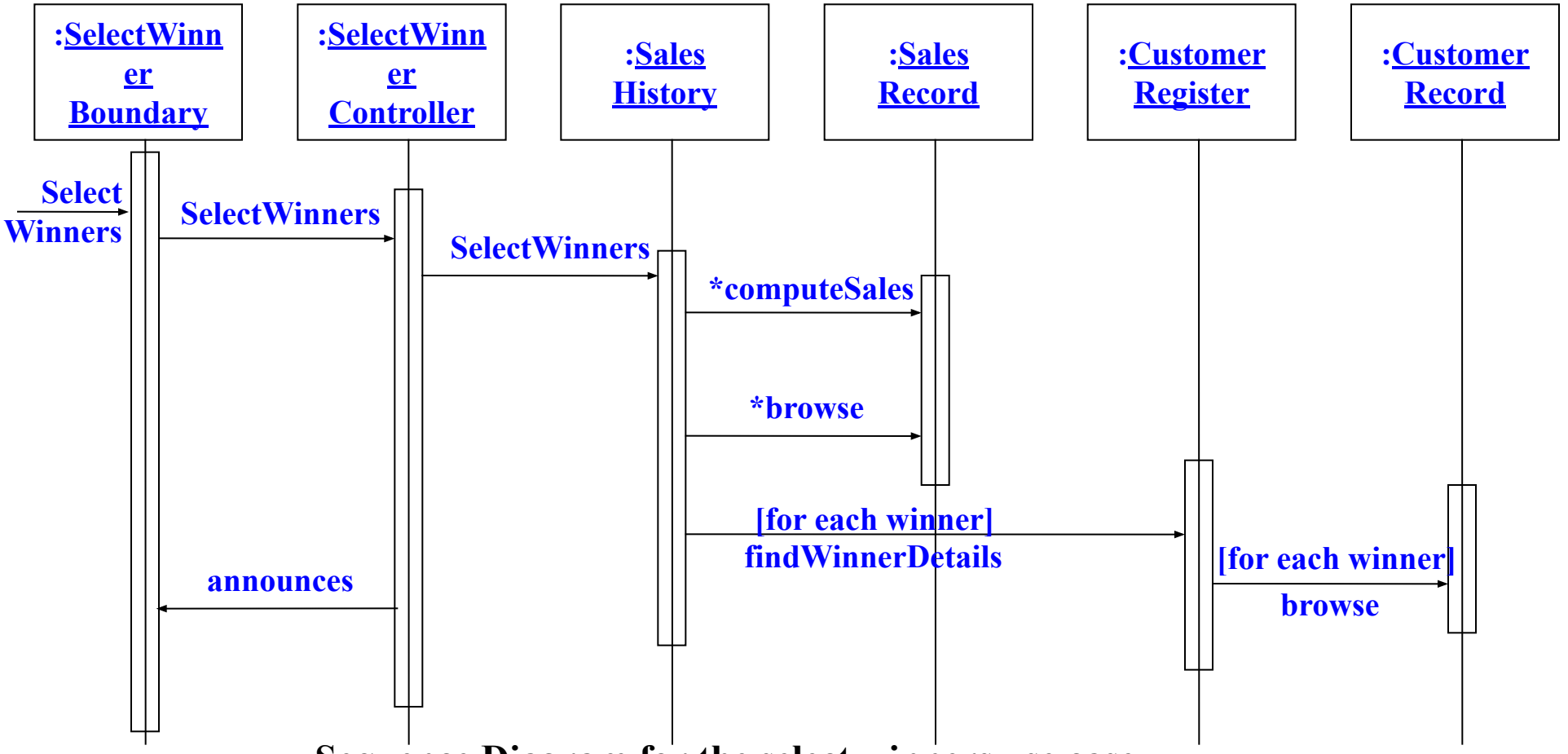
Sequence Diagram for the register customer use case

Example 2: Sequence Diagram for the Register Sales Use Case



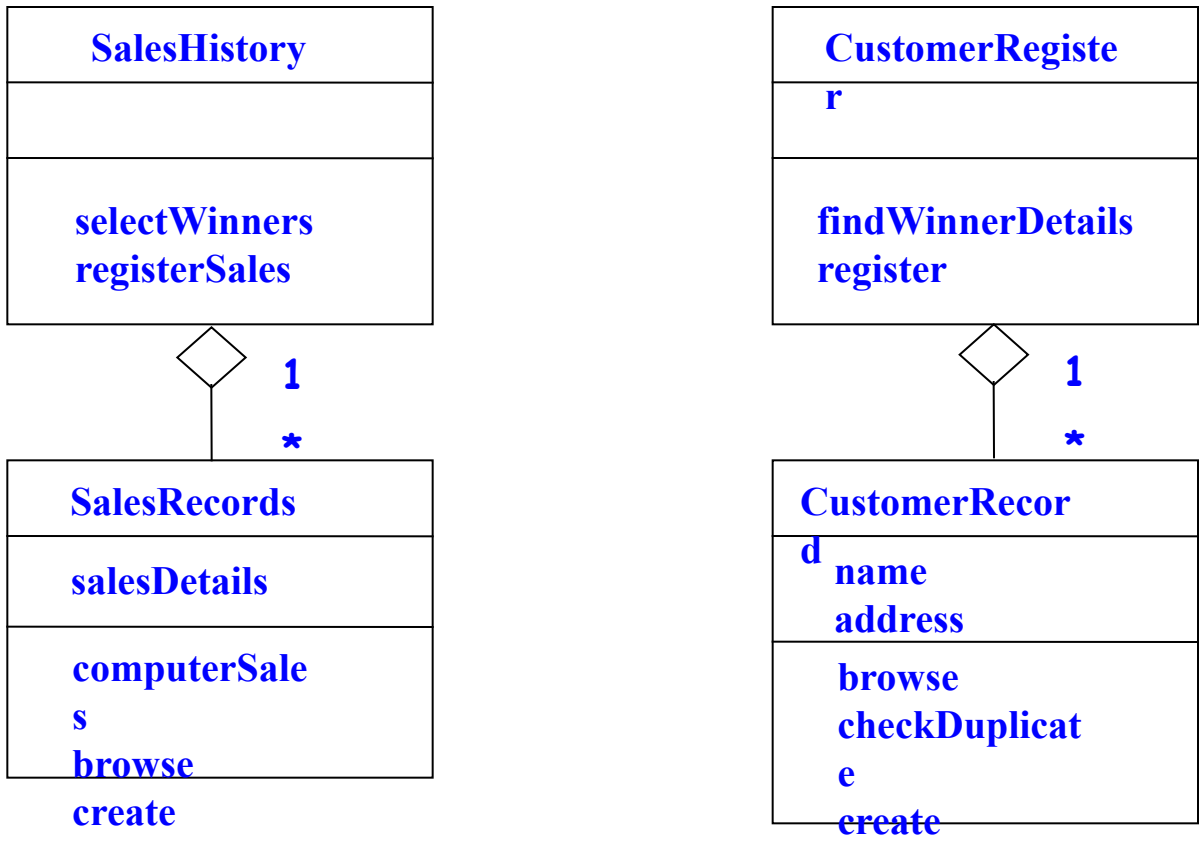
Sequence Diagram for the register sales use case

Example 2: Sequence Diagram for the Select Winners Use Case



Sequence Diagram for the select winners use case

Example 2: Class Diagram



CRC Card

Used to assign responsibilities (methods) to classes.

Complex use cases:

- Realized through collaborative actions of dozens of classes.
- Without CRC cards, it becomes difficult to determine which class should have what responsibility.

Class-Responsibility-Collaborator(CRC) Cards

- Pioneered by Ward Cunningham and Kent Beck.
- Index cards prepared one each per class.
- Contains columns for:
  - Class responsibility
  - Collaborating objects

CRC Cards Cont…

Systematize development of interaction diagram of complex use cases.

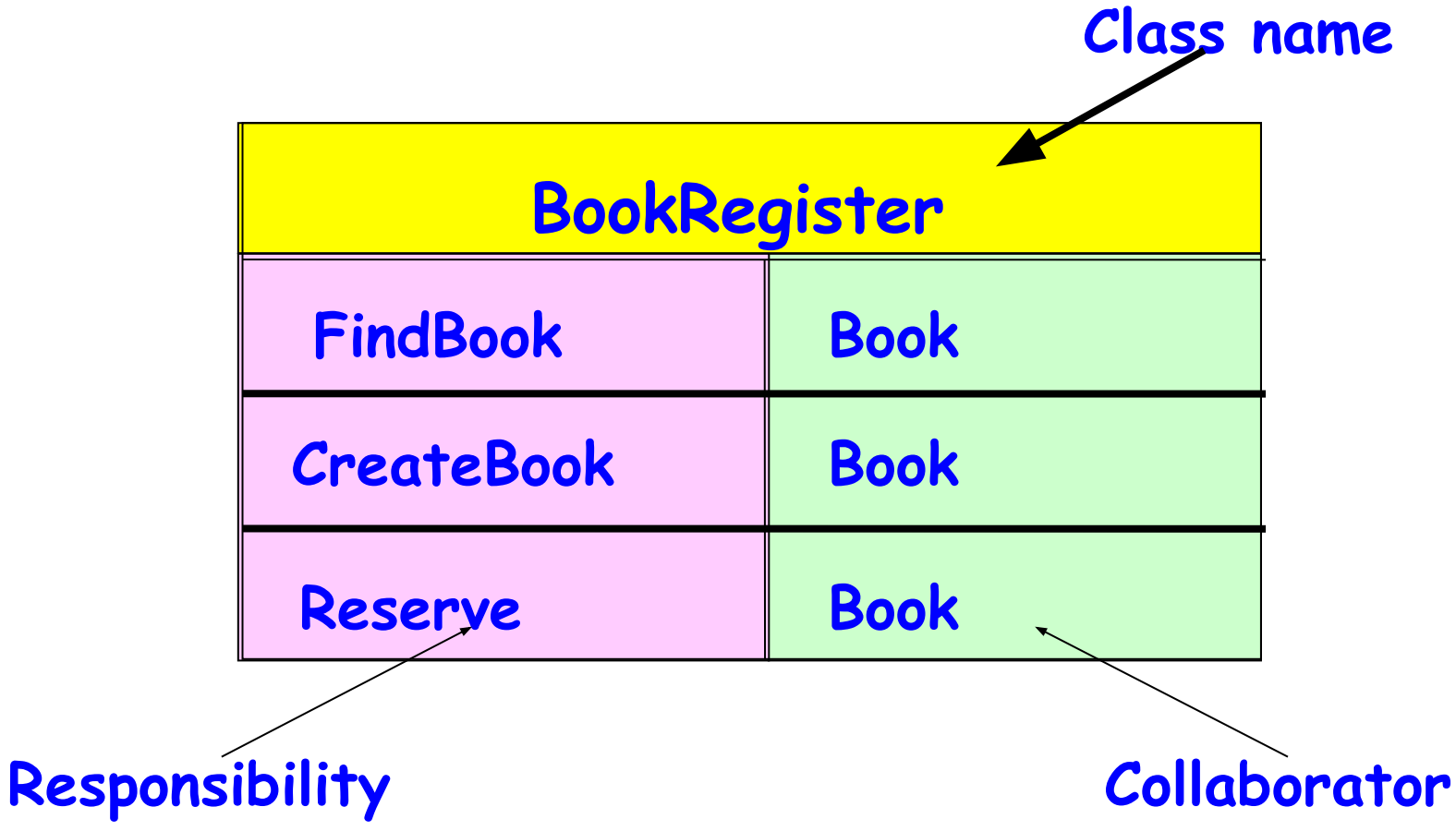
Team members participate to determine:

- The responsibility of classes involved in a use case realization

CRC Cards Cont…

- **Responsibility:**
  - Method to be supported by the class.
- **Collaborator:**
  - Class whose service (method) would have to be invoked

An Example of A CRC Card



CRC card for the BookRegister class

Using CRC Cards

- After developing a set of CRC cards:
  - Run simulations
  - Also known as structured walkthrough scenarios
- Execute a scenario :
  - A class is responsible to perform some responsibilities
  - It may then pass control to a collaborator -- another class
  - You may discover missing responsibilities and classes