CS-3004
SOFTWARE DESIGN AND ANALYSIS

RUBAB JAFFAR

RUBAB.JAFFAR@NU.EDU.PK

Robustness analysis Diagram

Lecture # 13,14,15

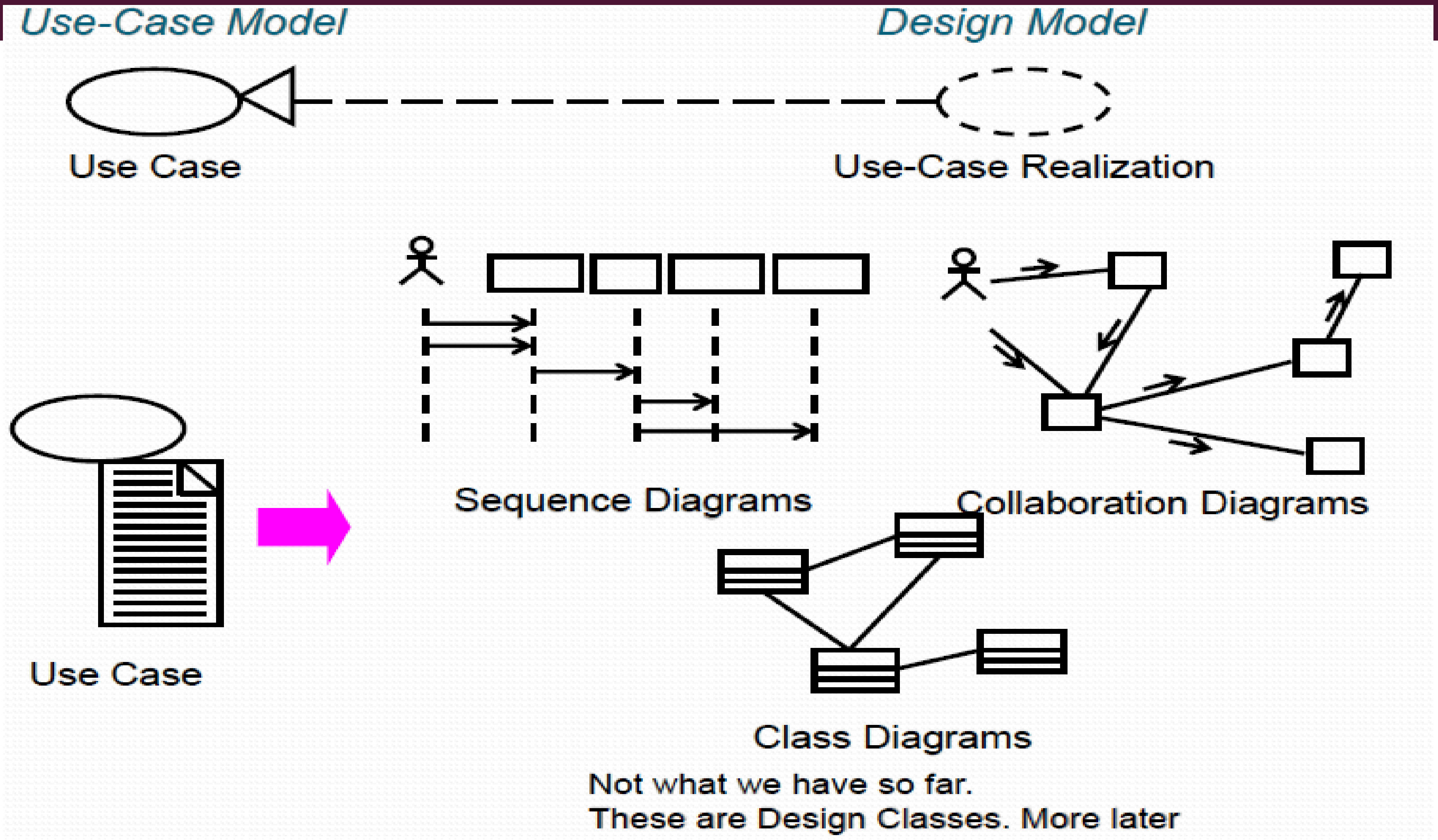
TODAY'S OUTLINE

- Class Diagram Exercises
- Use-case Realization
- Analysis Classes

USE CASE REALIZATION

- use-case realization describes how a particular use case is realized within the Design Model, in terms of collaborating objects.
- The diagrams that may be used to realize a use case realization may include:
- Interaction Diagrams (sequence and/or collaboration diagrams) can be used to describe how the use case is realized in terms of collaborating objects.
    - These diagrams model the detailed collaborations of the use-case realization.
- Class Diagrams can be used to describe the classes that participate in the realization of the use case, as well as their supporting relationships.
    - These diagrams model the context of the use-case realization.

USE CASE REALIZATION



Not what we have so far.
These are Design Classes. More later

WHO DOES USE CASE REALIZATION?

- A designer: responsible for the integrity of the use-case realization.
- Must coordinate with the designers responsible for the classes and relationships employed in the use-case realization.
- The use-case realization can be used by class designers to understand the class's role in the use case and how the class interacts with other classes.
  - This implies that a team will/may distribute responsibilities for each class to developers.
- This information can be used to determine/refine the class responsibilities and interfaces.
- Let's find the classes from different behaviors the classes must provide…

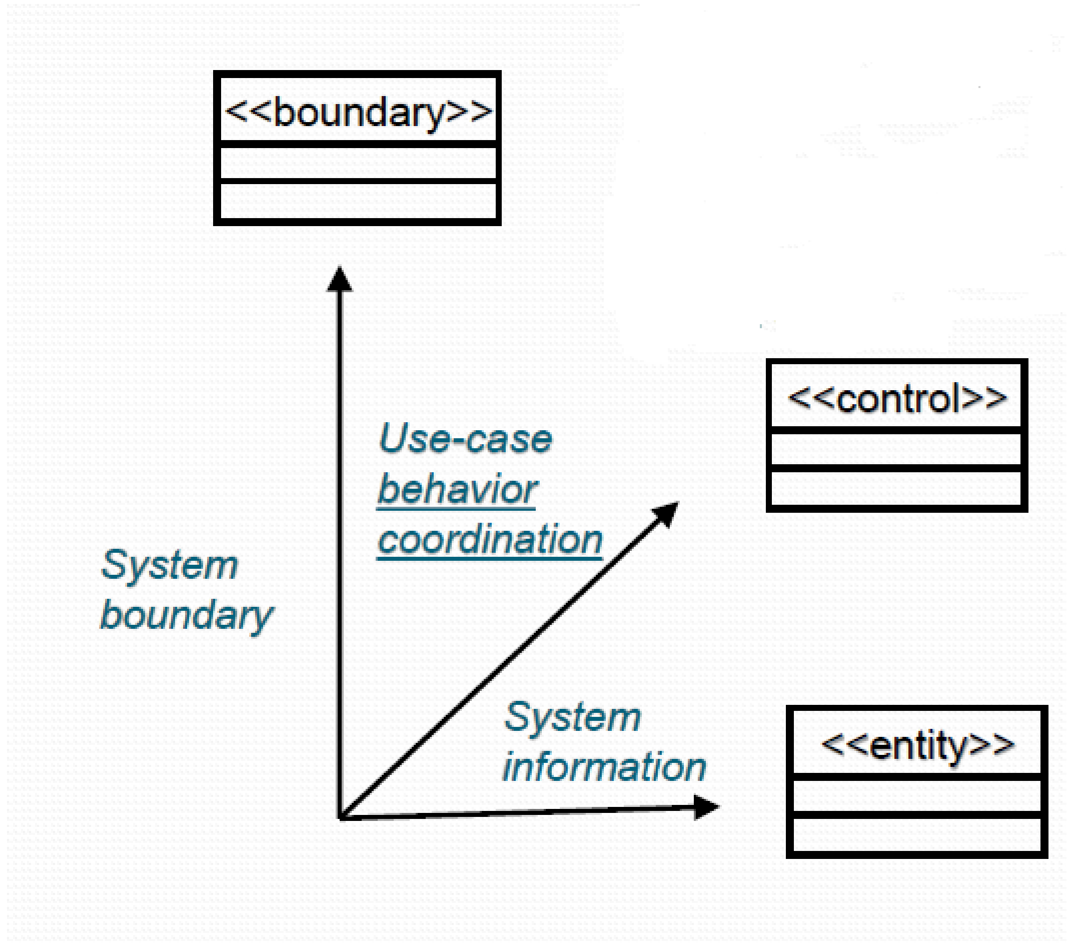
ANALYSIS CLASSES - EARLY CONCEPTUAL MODEL

- **The analysis classes, taken together, represent an early conceptual model of the system.**
- **This conceptual model evolves quickly and remains fluid for some time as different representations and their implications are explored.**
- **Analysis classes are early estimations of the composition of the system; they rarely survive intact into implementation.**
- **Many of the analysis classes morph into something else later on (subsystems, components, split classes, combined classes).**
- **They provide us with a way of capturing the required behaviors in a form that we can use to explore the behavior and composition of the system.**

IDENTIFYING CANDIDATE CLASSES FROM BEHAVIOR

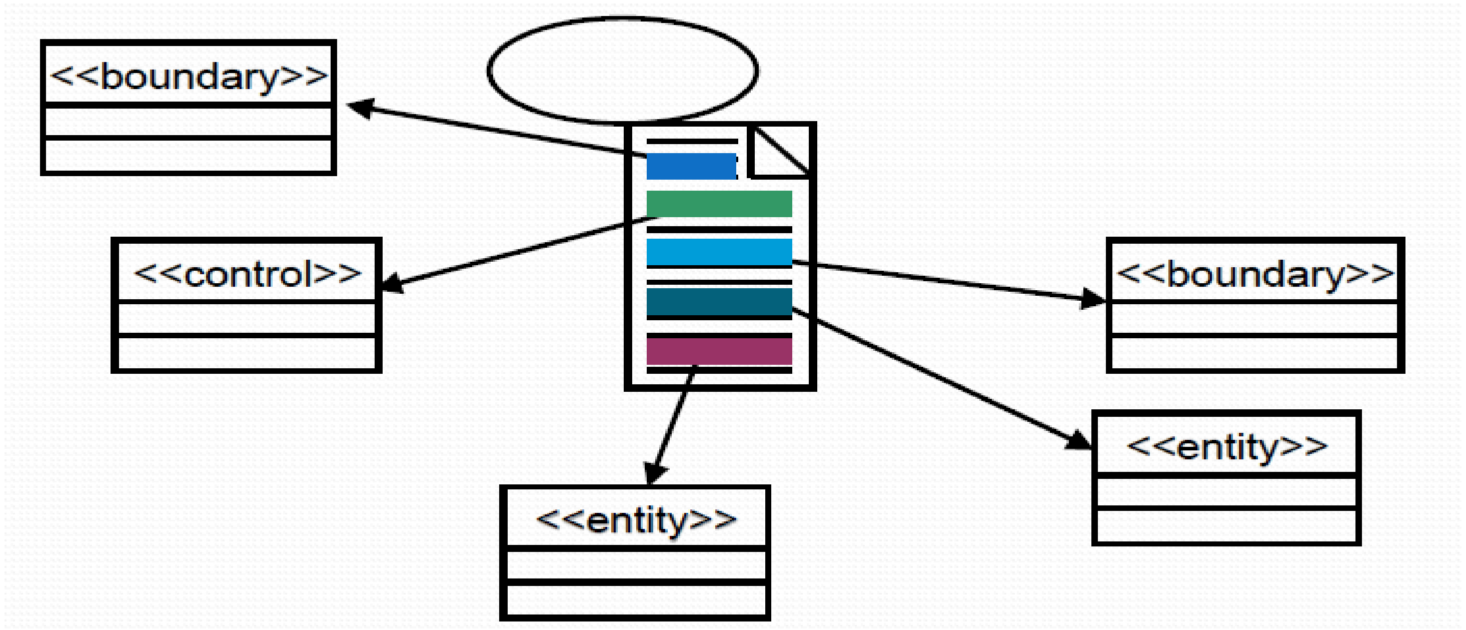
- Will use three perspectives of the system to identify these classes.
  - The ‘boundary’ between the system and its actors
  - The information’ the system uses
  - The ‘control logic’ of the system
- Will use stereotypes to represent these perspectives (boundary, control, entity)
  - These are conveniences used during analysis that will disappear or be transitioned into different design elements during the design process.
- Will result in a more robust model because these are the three things that are most likely to change in system and so we isolate them so that we can treat them separately.
  - That is, the interface/boundary, the control, and the key system entities.….

WHAT IS AN ANALYSIS CLASS?

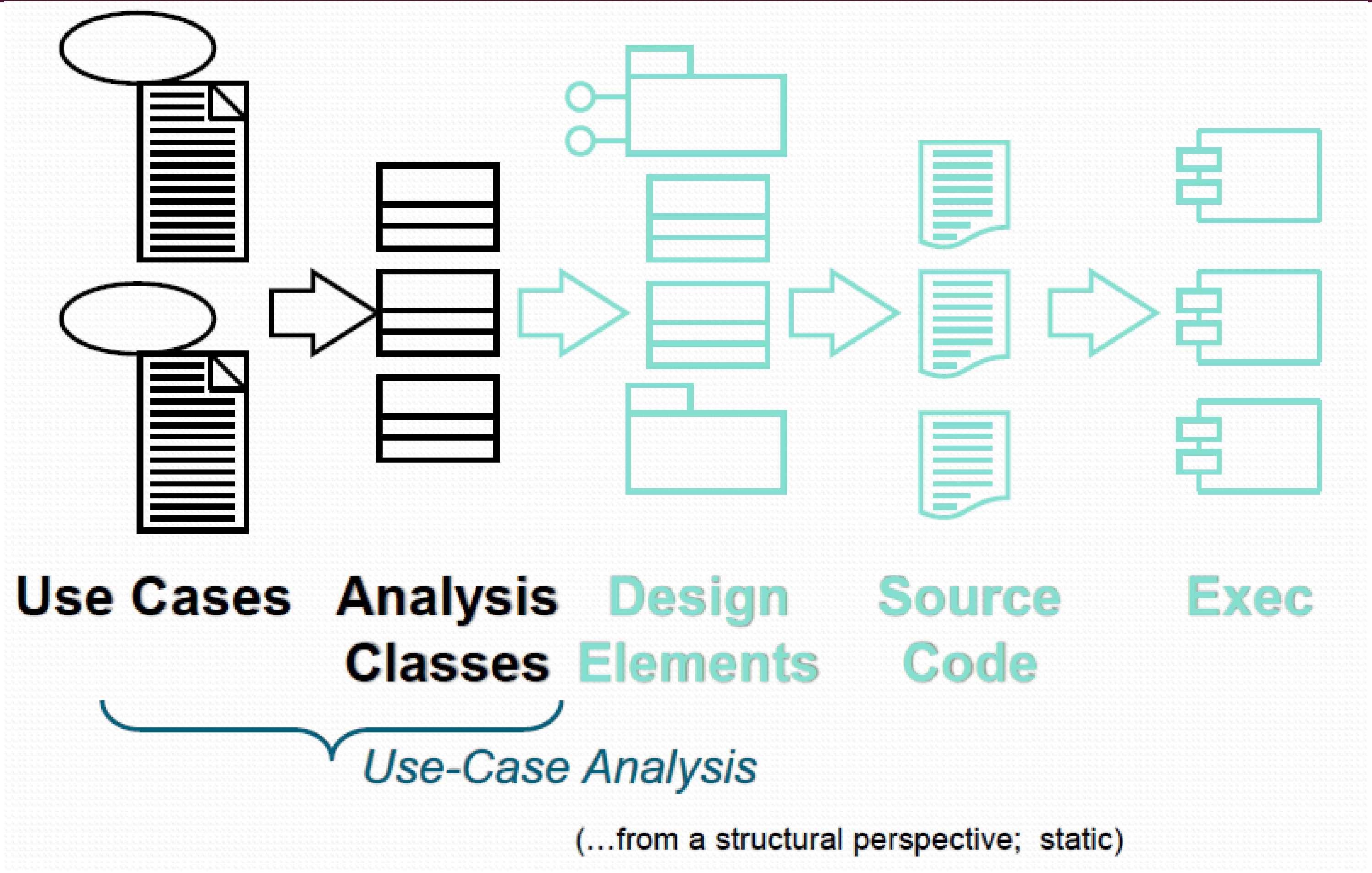


FIND CLASSES FROM USE-CASE BEHAVIOR

- **The complete behavior of a use case has to be distributed to analysis classes**
- **We must 'identify' these classes – identify, name, and briefly describe in a few sentences.**



ANALYSIS CLASSES: A FIRST STEP TOWARDS EXECUTABLES



DISCOVERING CLASSES

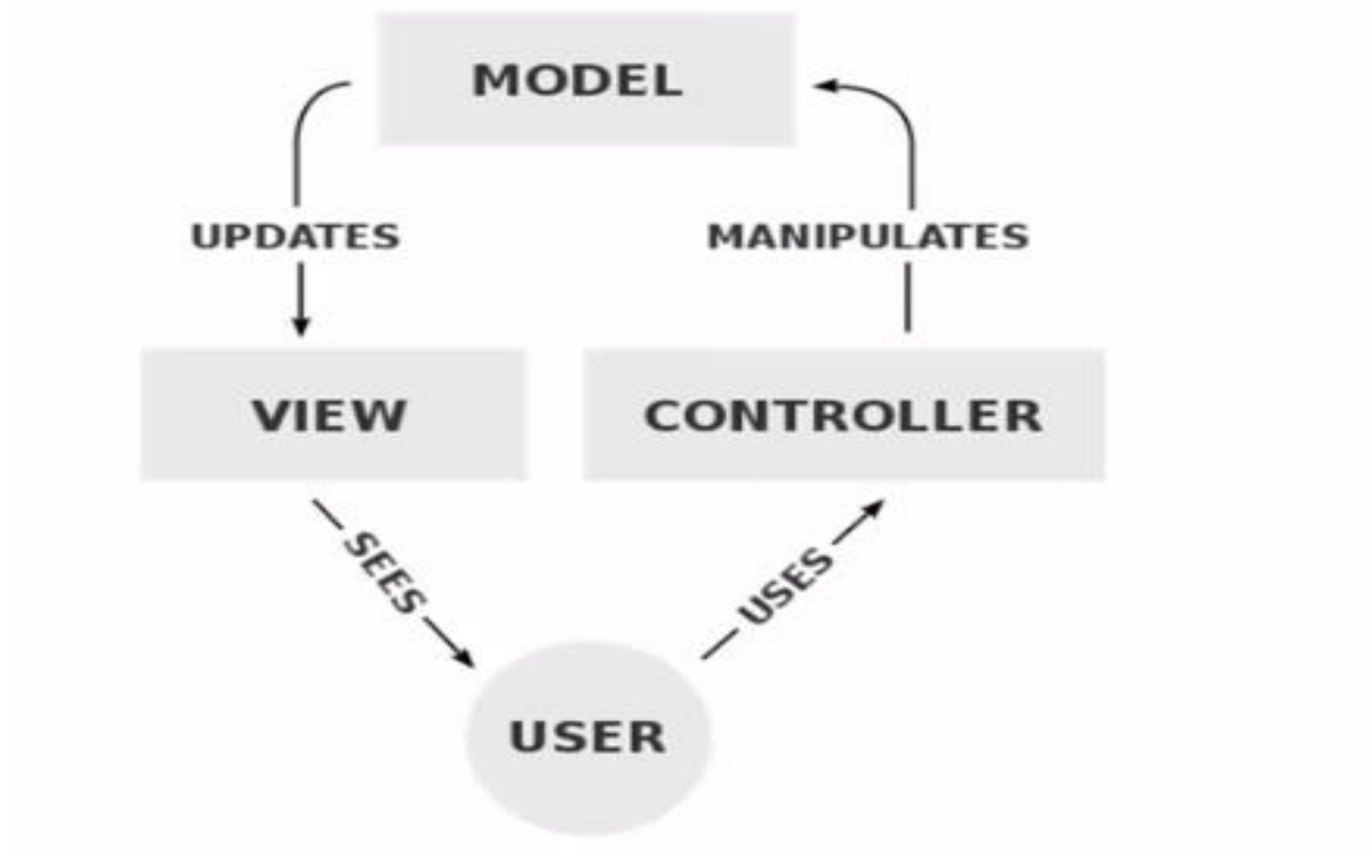
- **Analysis classes represent an early conceptual model for ‘things in the system which have responsibilities and behaviors’.**
- **Analysis classes are used to capture a ‘first-draft’, rough-cut of the object model of the system.**
- **Analysis classes handle primary functional requirements, interface requirements, and some control - and model these objects from the problem domain perspective.**

ENTITY , CONTROL, AND BOUNDARY DESIGN PATTERN

- **The ECB pattern represents a refinement of the model-view-controller (MVC) design pattern, a pattern that dates back to the early days of the Smalltalk-80 object-oriented language**

- **The goal of the MVC design pattern is to decompose the application into three distinct types of objects:**
  - **Model Objects: It expresses the application's behavior**
  - **View Objects: any output representation of information**
  - **Controller Objects: accepts input and converts it to commands for the model or view**

MVC PATTERN



ENTITY, CONTROL, AND BOUNDARY DESIGN PATTERN

- **The ECB design pattern is closely related to the MVC design pattern**

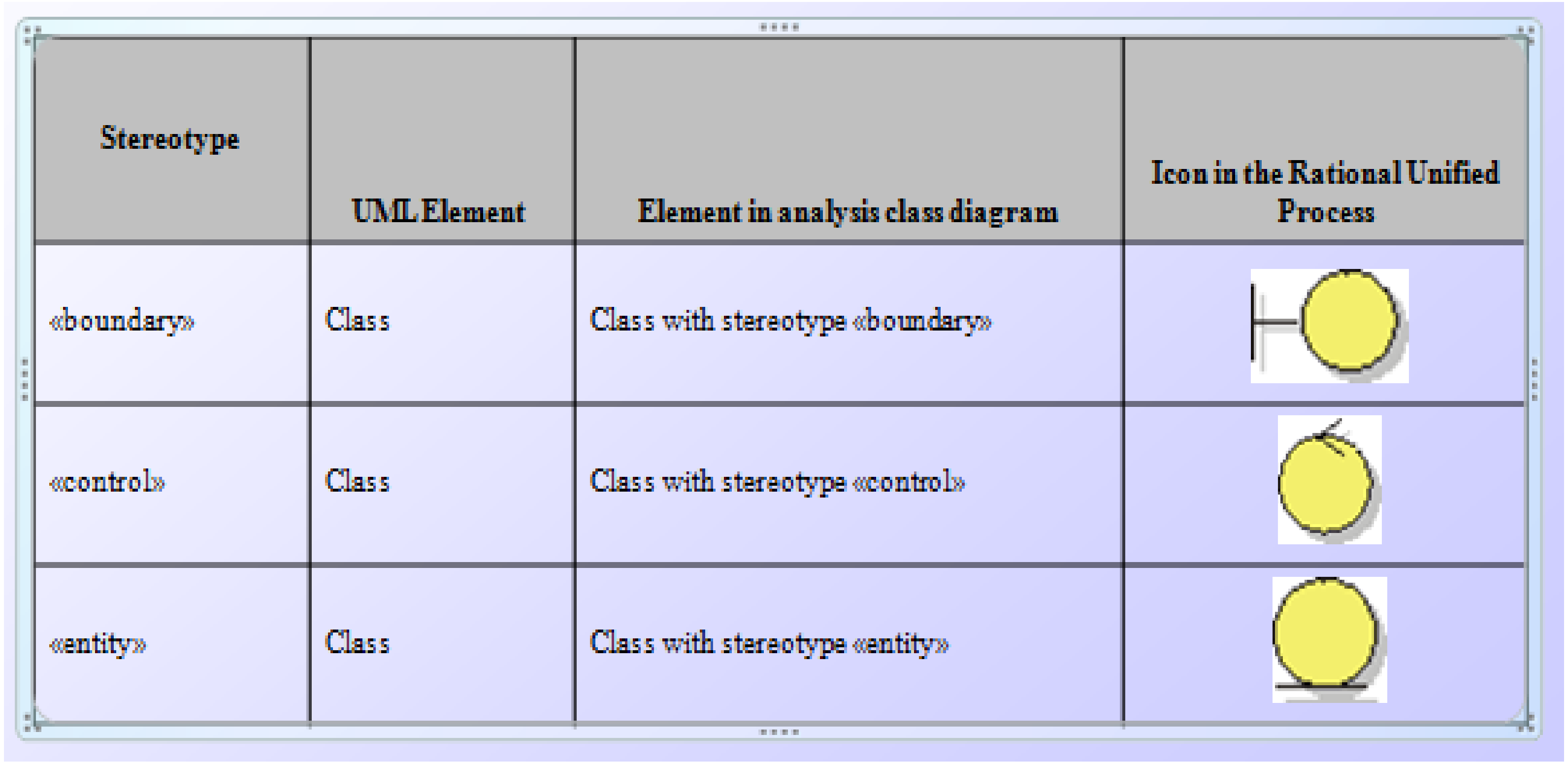
- **As such, its goal is to decompose the application into three distinct types of objects:**
  - **Boundary objects**
  - **Control objects**
  - **Entity objects**
- **Rules govern how each of these objects can communicate with the other objects associated with the pattern**

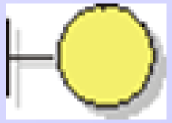
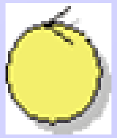
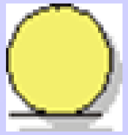
- **The primary distinction between these two design patterns is the rules that govern object communication**

ENTITY, CONTROL, AND BOUNDARY DESIGN PATTERN

- **This innovative approach to analysis and design was originally introduced by Doug Rosenberg and Kendall Scott**
- **Stereotypes (i.e., a meta-classification of an element) based upon these three object types are available in UML packages for creating interaction and class diagrams**
- **Once this pattern has been described, an analysis or design model will become very recognizable**

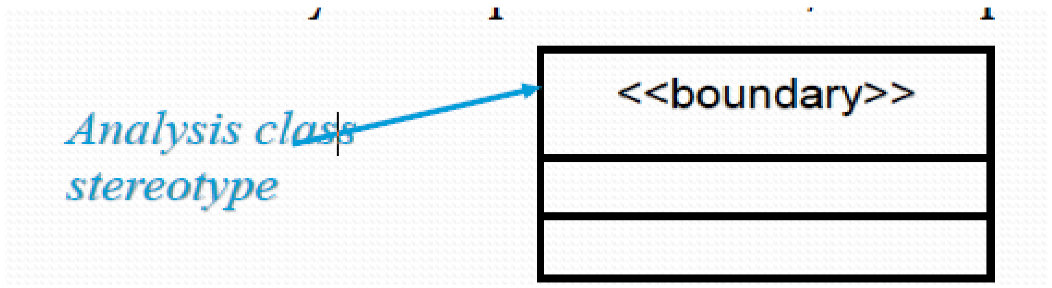
ENTITY, CONTROL, AND BOUNDARY DESIGN PATTERN



Stereotype	UML Element	Element in analysis class diagram	Icon in the Rational Unified Process
«boundary»	Class	Class with stereotype «boundary»	
«control»	Class	Class with stereotype «control»	
«entity»	Class	Class with stereotype «entity»	

BOUNDARY OBJECTS

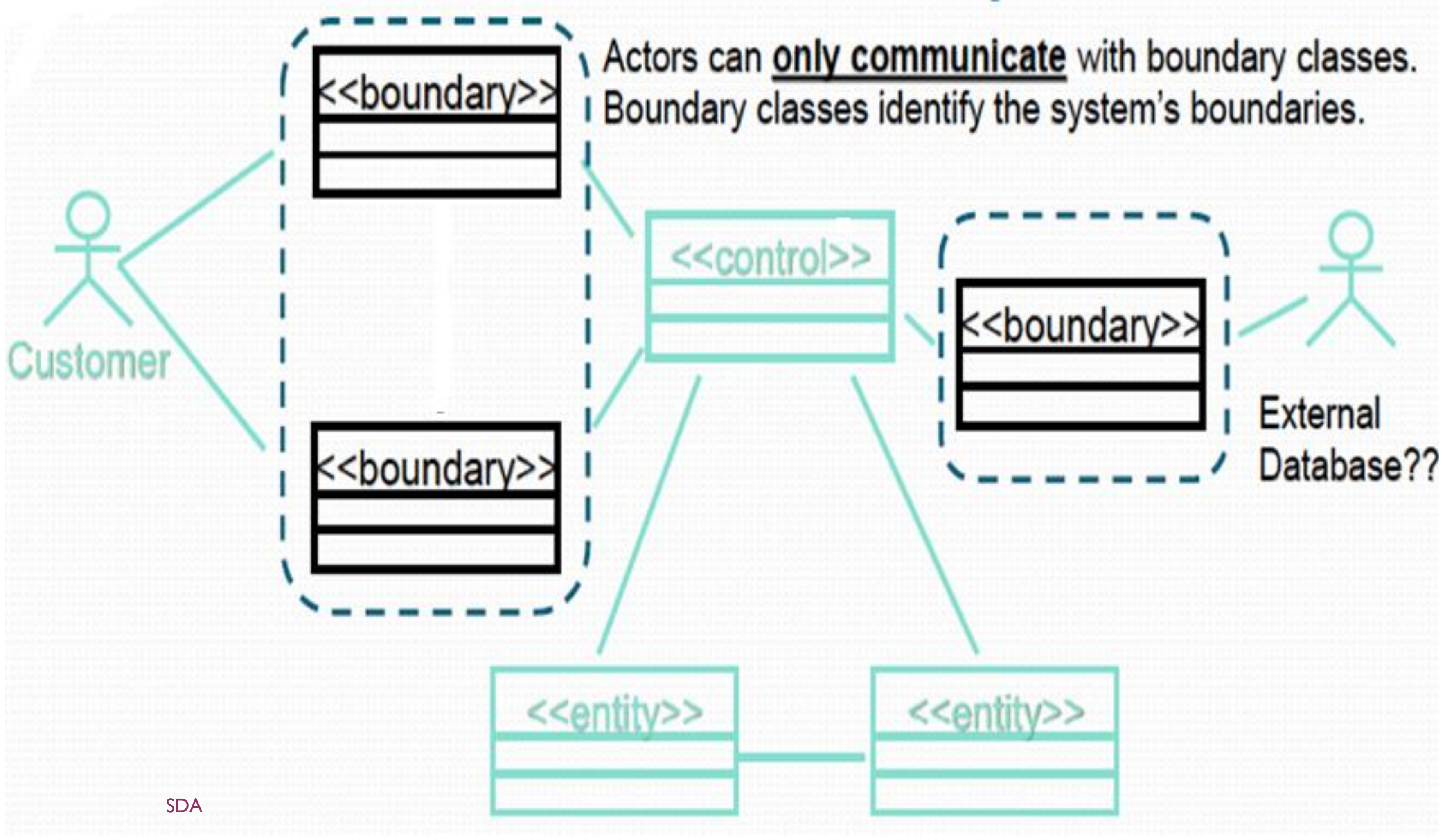
- **Boundary objects are responsible for supporting communications between the system's external environment and its internal workings (i.e., control and entity objects)**
- **Within the context of use case realization, there will be one boundary class for each user interface**
- **The actor(s) identified within the Use Case Model will always interact with the system through these boundary objects**



BOUNDARY CLASS

- **Insulates the system from changes in the outside**
- **Several Types of Boundary Classes**
- **User interface classes – classes that facilitate communication with human users of the system**
  - **Menus, forms, etc. User interface classes….**
- **System interface classes – classes which facilitate communications with other systems.**
  - **These boundary classes are responsible for managing the dialogue with the external system, like getting data from an existing database system or flat file…**
  - **Provides an interface to that system for this system**
- **Device Interface Classes – provide an interface to devices which detect external events – like a sensor**

THE ROLE OF BOUNDARY CLASS



Actors can only communicate with boundary classes.
Boundary classes identify the system's boundaries.
<<boundary>>
<<boundary>>
Customer
<<control>>
<<boundary>>
External Database??
<<entity>>
<<entity>>

BOUNDARY CLASS- MORE

- **Identify boundary classes for things mentioned in the flow of events of the use-case realization.**
- **Consider the source for all external events and make sure there is a way for the system to detect these events. (user inputs/responses? Connection to existing external data…)**
- **One recommendation: for the initial identification of boundary classes is one boundary class per actor/use-case pair.**
  - **This class can be viewed as having responsibility for coordinating the interaction with the actor.**
  - **This may be refined as a more detailed analysis is performed.**
  - **This is particularly true for window-based GUI applications, where there is typically one boundary class for each window, or one for each dialog.**

CONTROL CLASSES

- Control objects are responsible for application specific business logic
- In addition, these object types also function as an intermediary between the system’s various boundary and entity objects
- Within the context of use case realization, each boundary class will communicate with a single control class and control classes will be used to manage each use case’s flow of execution
- To manage this flow, the control object must coordinate the activities required to support the use case realization, including interactions with other control objects and the data aware entity objects
- Each entity object will be tightly coupled with a control object

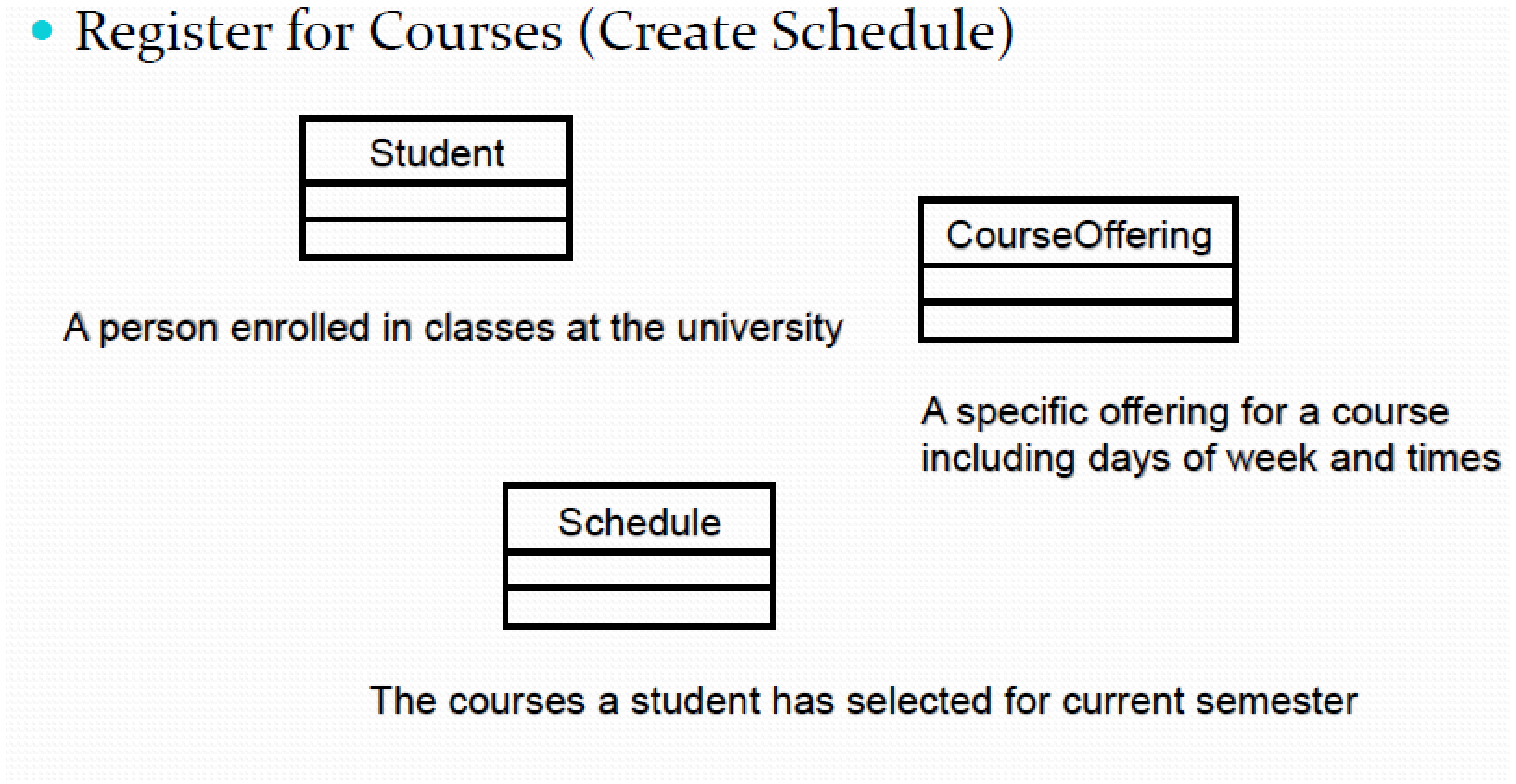
ENTITY CLASSES

- Entity classes represent stores of information in the system
- They are typically used to represent the key items the system manages.
- Entity objects (instances of entity classes) are used to hold and update information about some phenomenon, such as an event, a person, or some real-life object.
  - (advisor, teacher, university, student etc.)
- They are usually persistent, having attributes and relationships needed for a long period, sometimes for the life of the system.
- The main responsibilities of entity classes are to store and manage information in the system.
- To display the data, the data encapsulated within the entity object will traverse a path that eventually leads to the control object that is tightly coupled to the boundary object
- At this point, the data will be passed to the boundary object for displaying in the GUI

CANDIDATE ENTITY CLASSES

- Sometimes there is a need to model information about an actor within the system. This is not the same as modeling the actor (actors are external by definition).
- For example, a course registration system maintains information about the student which is independent of the fact that the student also plays a role as an actor of the system.
  - This information about the student that is stored in a 'Student' class is completely independent of the 'actor' role the student plays; the Student class (entity) will exist whether or not the student is an actor to the system.

EXAMPLE: CANDIDATE ENTITY CLASSES



- Register for Courses (Create Schedule)

A person enrolled in classes at the university

A specific offering for a course including days of week and times

The courses a student has selected for current semester

WHAT IS AN ANALYSIS CLASS?

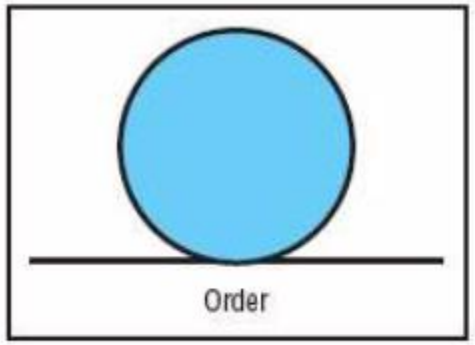
- **A class that represents initial data and behavior requirements, and whose software and hardware oriented details have not been specified**
- **Analysis class diagram – a UML diagram showing analysis classes and their relationships**

ANALYSIS CLASSES IN A NUT SHELL

- **Entity class :**
- **Persistent data**
- **used multiple times and in many UCs)**
- **Still exists after the UC terminates (e.g. DB storage)**
- **Boundary class:**
- **(User) interface between actors and the system**
- **E.g. a Form, a Window (Pane)**
- **Control class:**
- **Encapsulates business functionality**
- **Proposed in RUP (Rational Unified Process)**

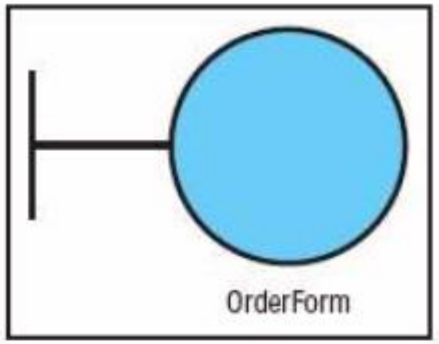
STEREOTYPES OF ANALYSIS CLASSES

Figure 9.3a
Entity Class Order



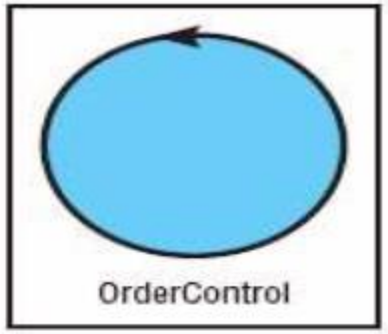
Mostly corresponds to conceptual data model classes

Figure 9.3b
Boundary Class
OrderForm



Encapsulates connections between actors and use cases

Figure 9.3c
Control Class
OrderControl



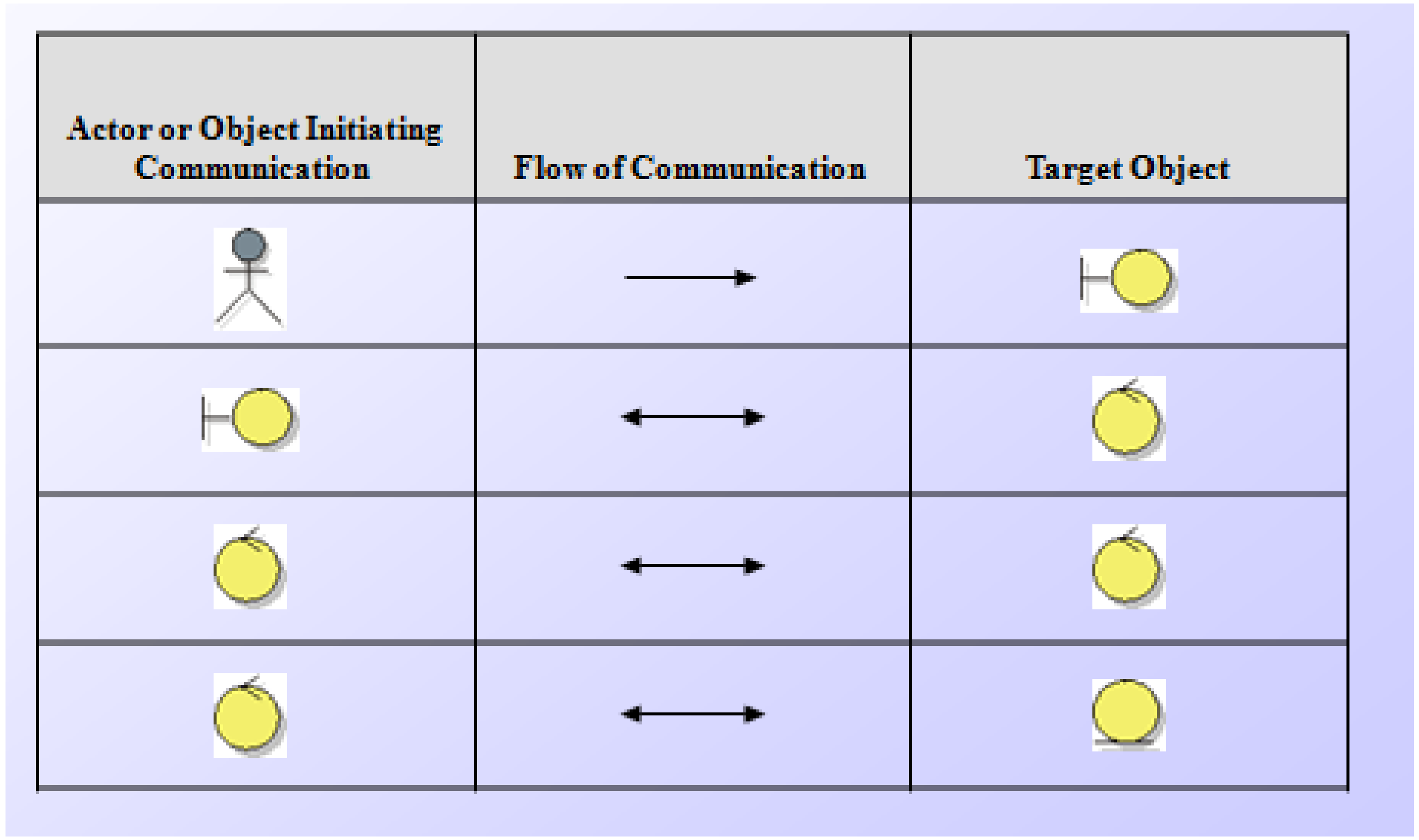
Mostly performs behaviors associated with inner workings of use cases

ECB PATTERN

- **Collaborating together, the various boundary, control, and entity objects within the BCE design pattern realize the behavior documented in the system's Use Case Model**

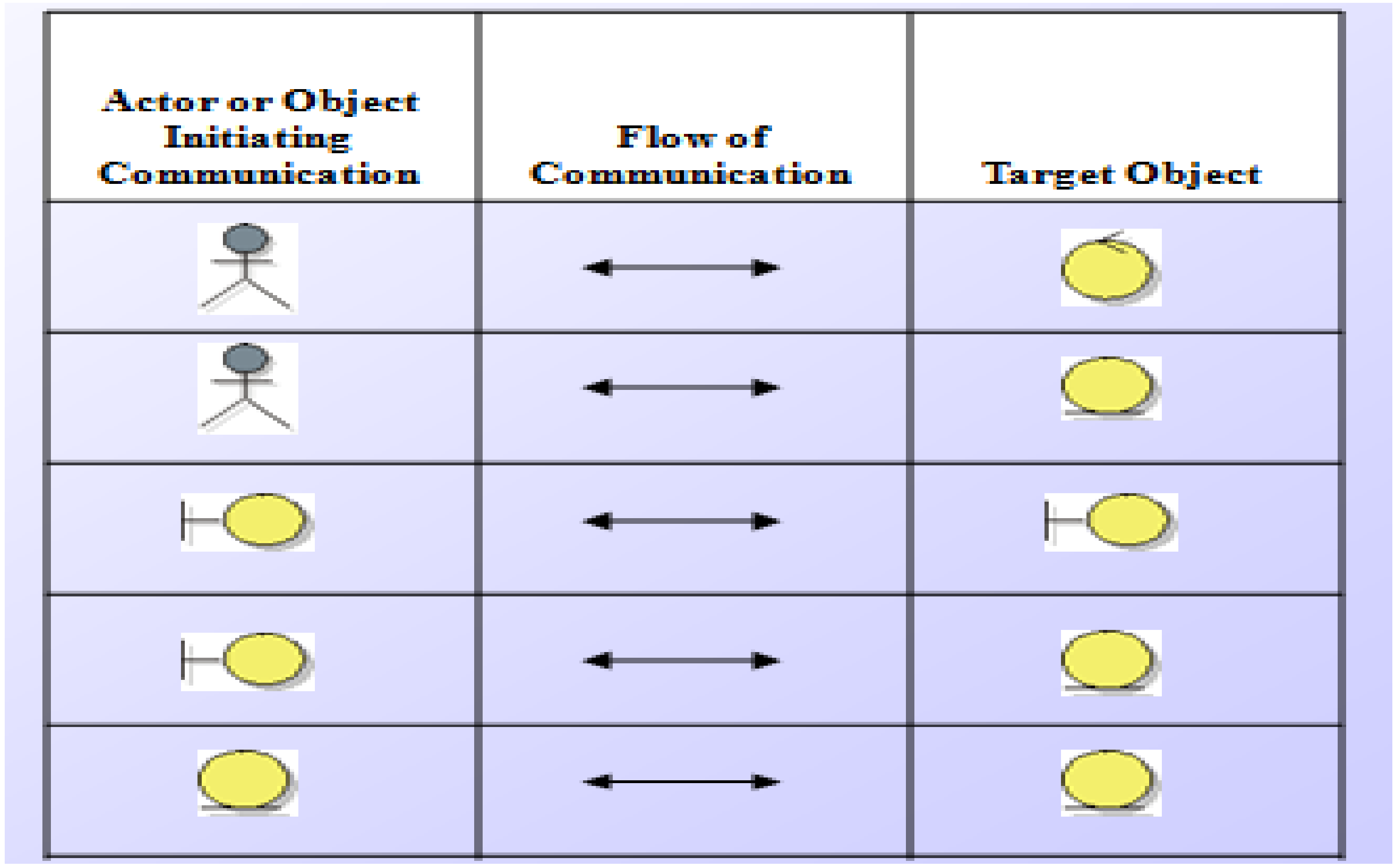
- **The rules that govern communication between the various object types within the BCE design pattern are illustrated in the tables that follow**

ALLOWED COMMUNICATION WITHIN THE ECB DESIGN PATTERN

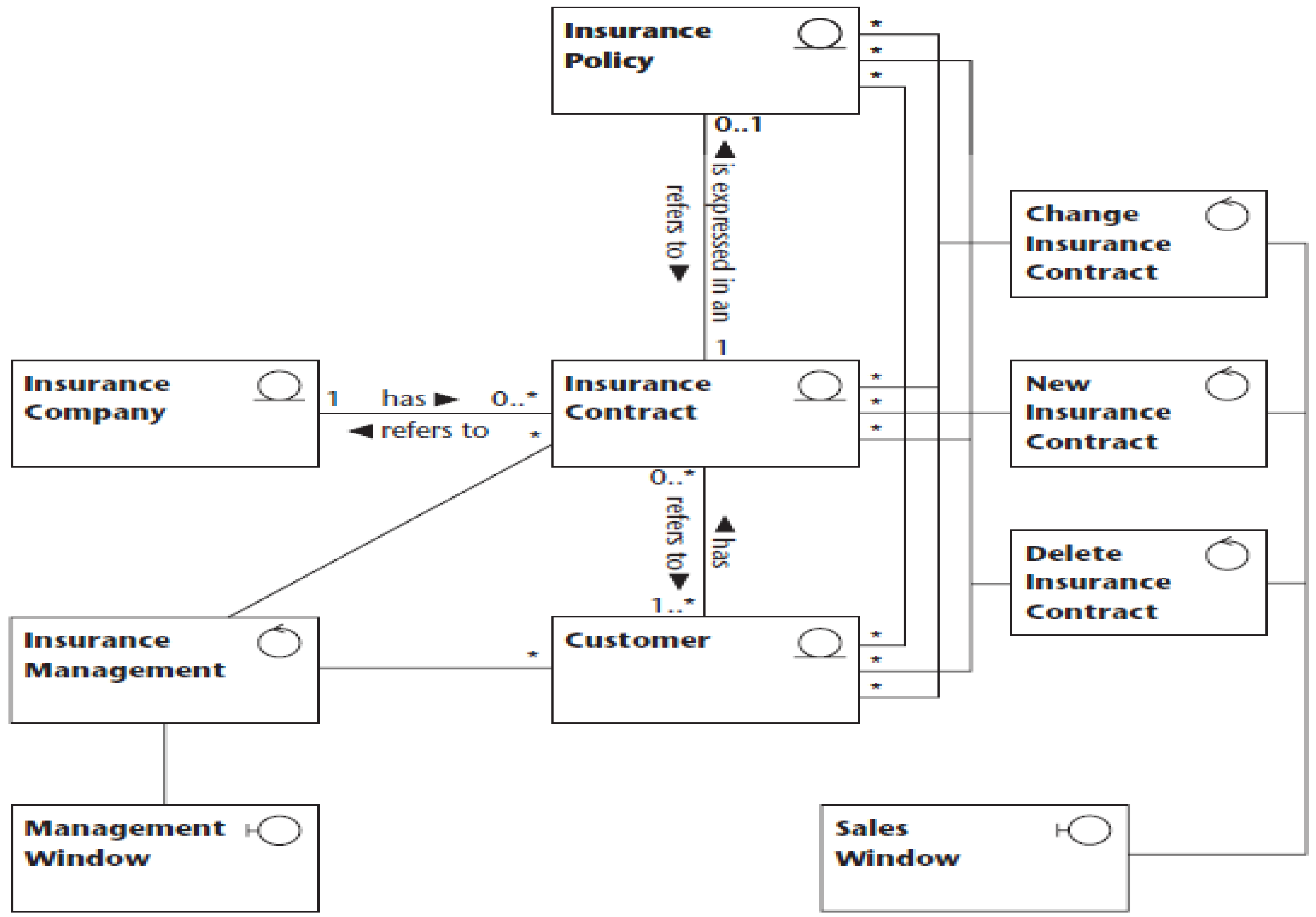


Actor or Object Initiating Communication	Flow of Communication	Target Object
🧍	→	⊢○
⊢○	↔	⟲
⟲	↔	⟲
⟲	↔	○̲

COMMUNICATION NOT ALLOWED WITHIN THE ECB

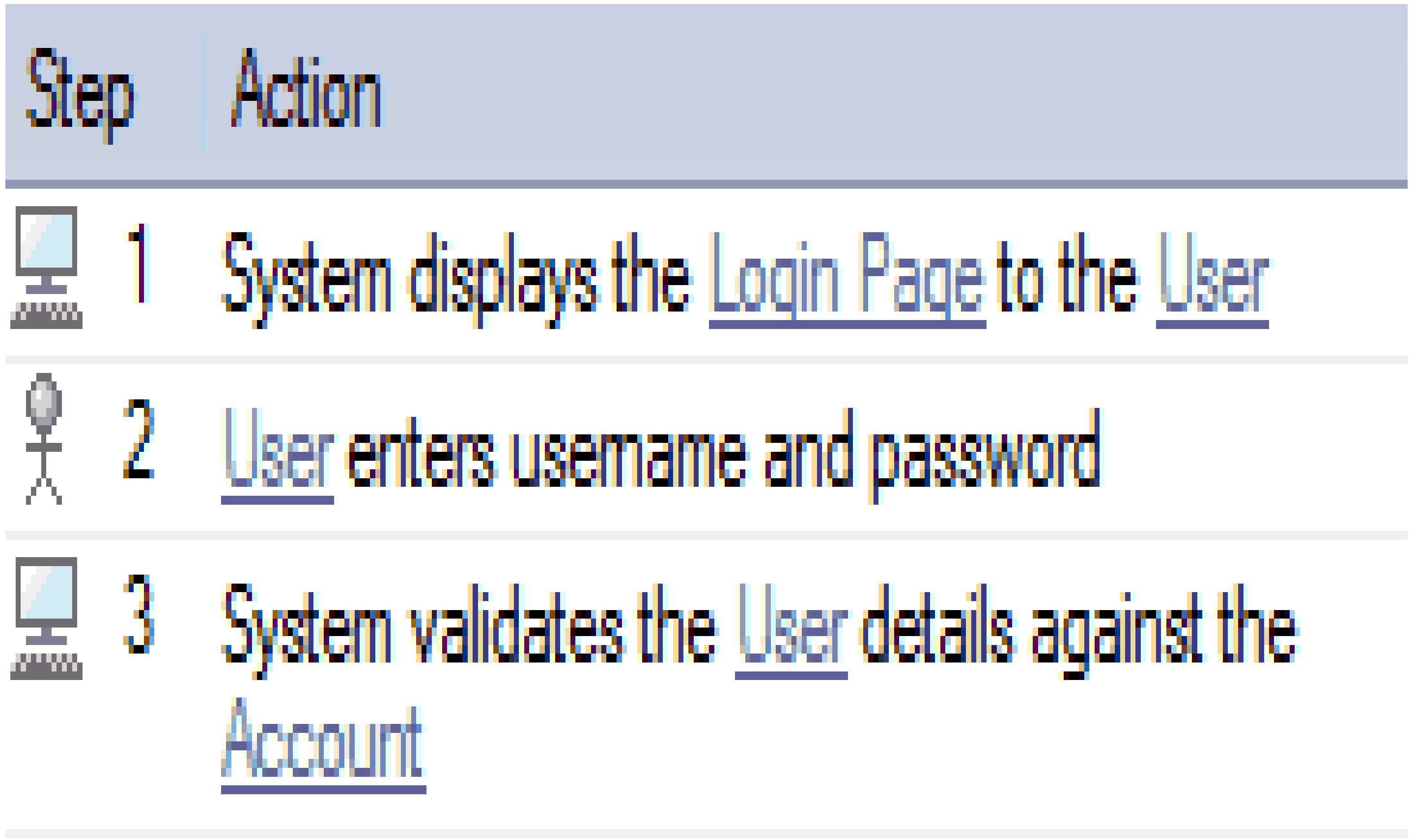


`

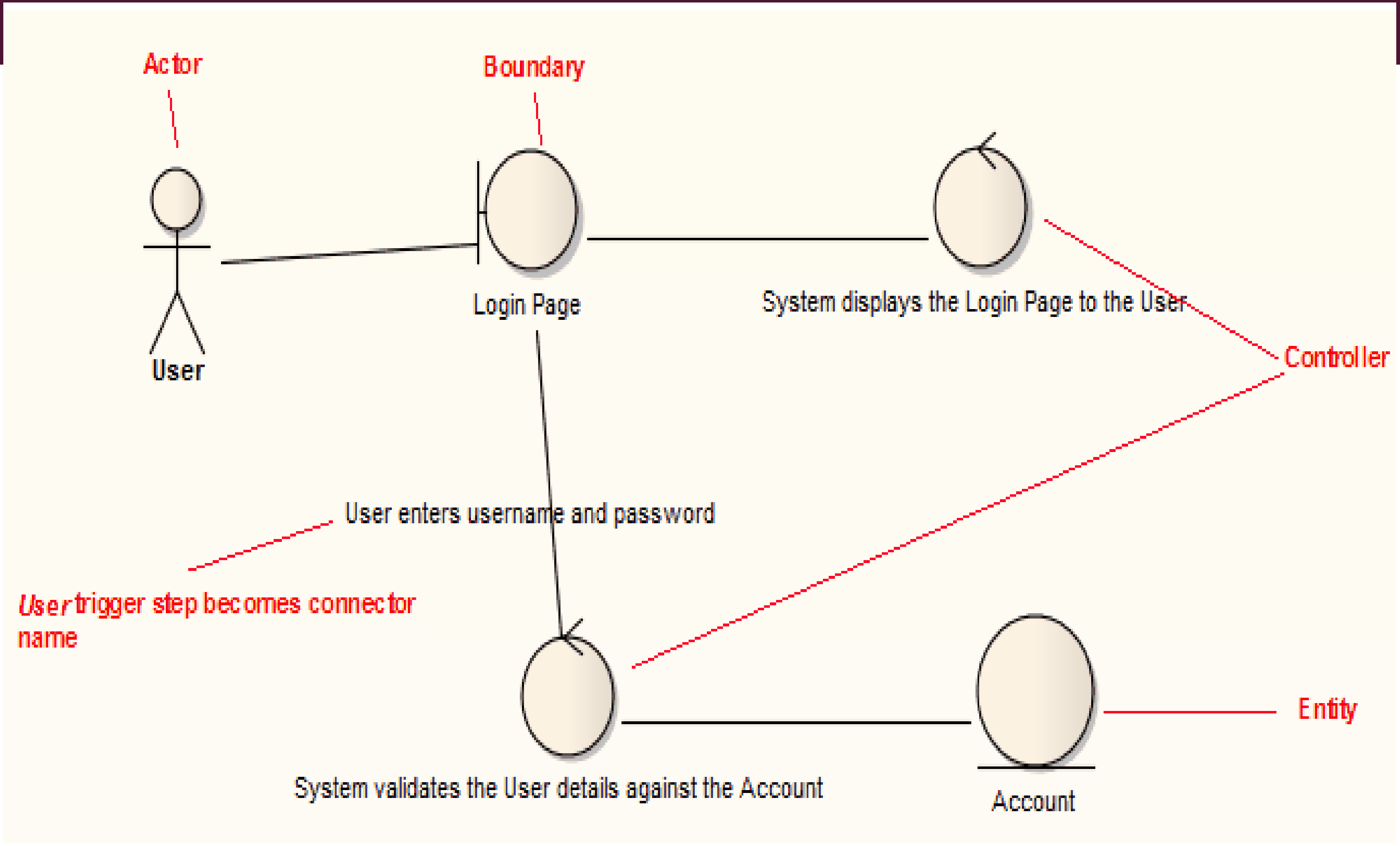


The entity classes capture the core business, in this case, the insurance business. The entity classes are Insurance Policy, Insurance Contract, Customer, and Insurance Company. The control classes Change Insurance Contract, New Insurance Contract, and Delete Insurance Contract serve the boundary class Sales Window. The control class Insurance Management serves the Management Window with Customers and Insurance Contracts.

ROBUSTNESS DIAGRAM ACTIVITY

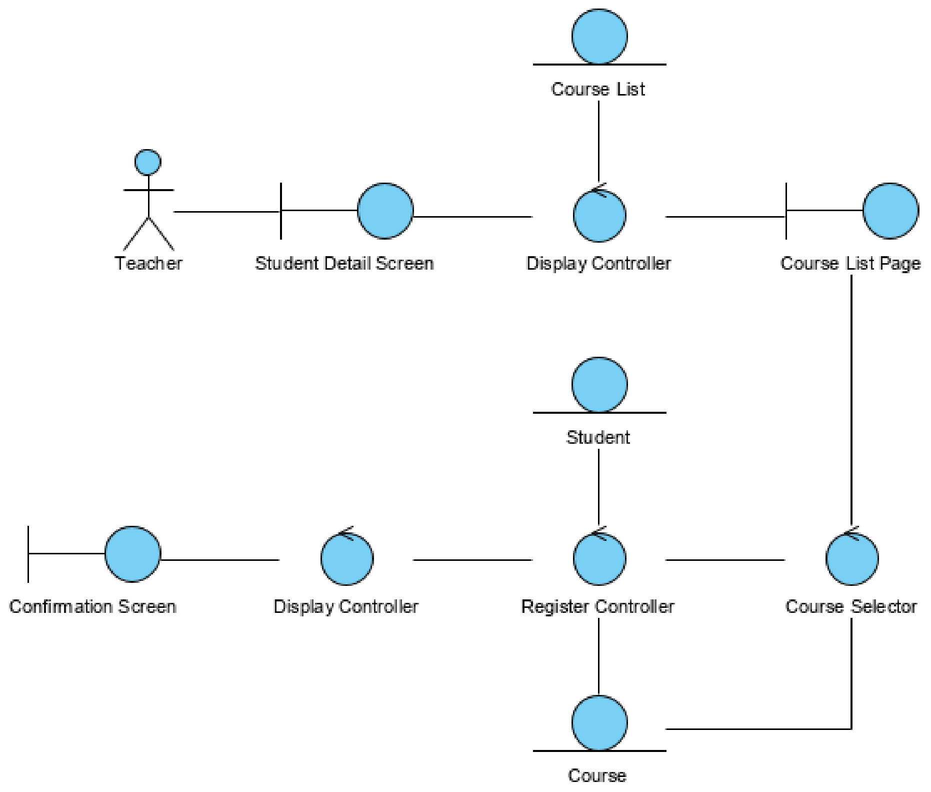


Step	Action
1	System displays the Login Page to the User
2	User enters username and password
3	System validates the User details against the Account


Actor
Boundary
User
Login Page
System displays the Login Page to the User
Controller
User enters username and password
User trigger step becomes connector name
System validates the User details against the Account
Account
Entity

YOUR TURN

- *From the student detail page, the teacher clicks on the "Add courses" button and the system displays the list of courses on the course list page. The teacher selects the name of a course and presses the "Register" button. The system registers the student for the course. And displays the confirmation message on Confirmation screen.*


Course List
Teacher
Student Detail Screen
Display Controller
Course List Page
Student
Confirmation Screen
Display Controller
Register Controller
Course Selector
Course

WHAT IS ROBUSTNESS ANALYSIS?

- Involves analyzing the narrative text of each of the use cases and identifying a first-guess set of the objects into entity, boundary, and control classes

- Requires completeness checks and adherence to diagramming rules

ROBUSTNESS DIAGRAM- ECB PATTERN(EXAMPLE)

- The **UNIVERSITY OF KARACHI** registration system is briefly described thus:
- You have been asked to streamline, improve, and automate the process of assigning professors to courses and the registration of students such that it takes advantage of prevailing web technologies for on-line real time, location independent access.
- The process begins by professors deciding on which courses they will teach for the semester. The Registrar's office then enters the information into the computer system, allocating times, room, and student population restrictions to each course. A batch report is then printed for the professors to indicate which courses they will teach. A course catalogue is also printed for distribution to students.

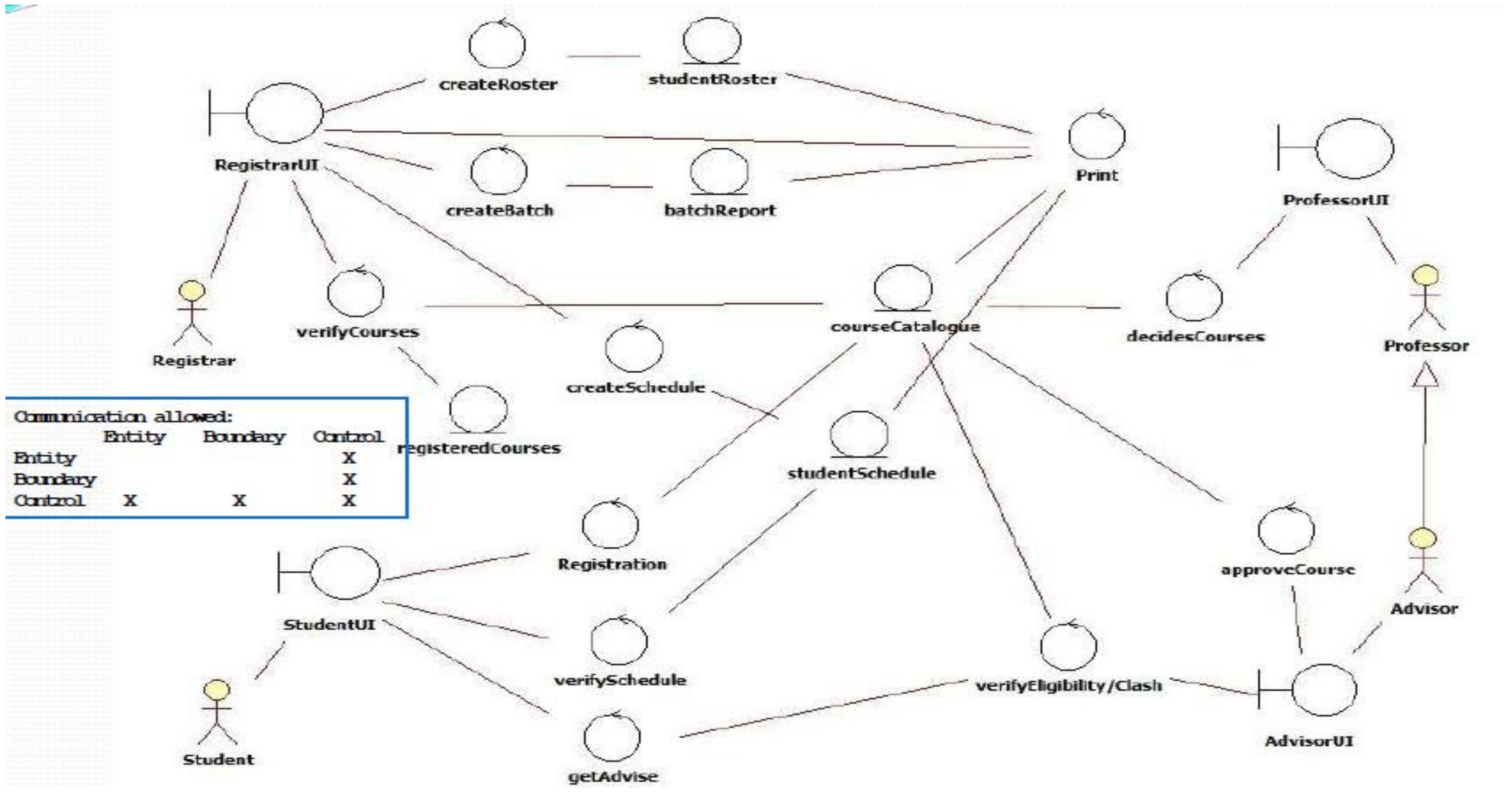
ROBUSTNESS DIAGRAM- ECB PATTERN(EXAMPLE)

- **Students then select what courses they desire to take and indicate these by completing paper-based course advising forms. They then meet with an academic advisor who verifies their eligibility to take the selected courses, that the sections of the courses selected are still open, and that the schedule of selected courses does not clash. The typical student load is four courses.**
-
- **The advisor then approves the courses and completed the course registration forms for the student. These are then sent to the registrar who keys them into the registration system – thereby formally registering a student. If courses selected are not approved, the student has to select other courses and complete the course advising forms afresh.**

ROBUSTNESS DIAGRAM- ECB PATTERN(EXAMPLE)

- Most times students get their first choice, however, in those cases where there is a conflict, the advising office talks with the students to get additional choices. Once all students have been successfully registered, a hard copy of the students' schedule is sent to the students for verification.
- Most student registrations are processed within a week, but some exceptional cases take up to two weeks to resolve.
- Once the initial registration period is over, professors receive a student roster for each class they are scheduled to teach.

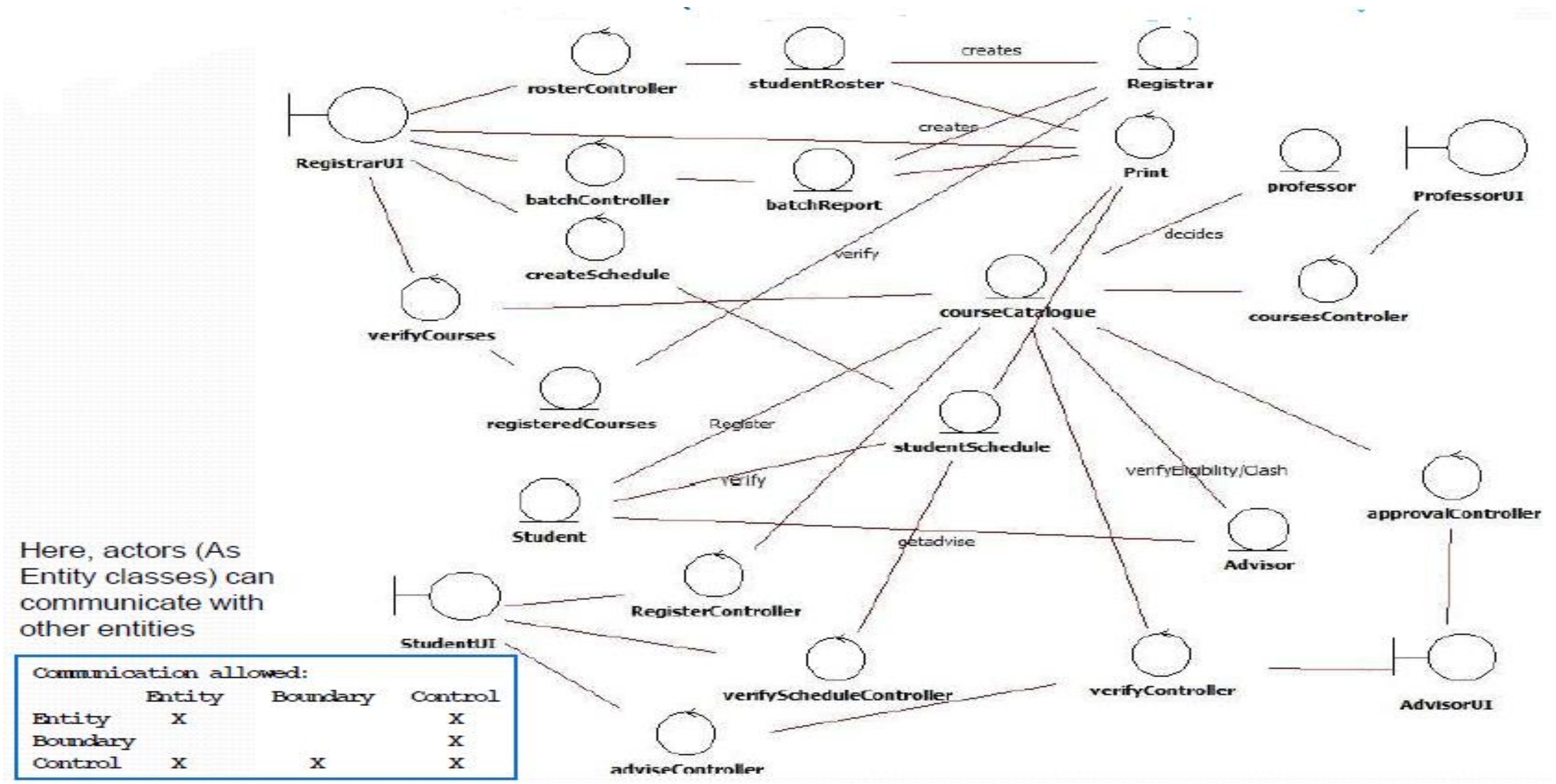
ROBUSTNESS DIAGRAM- ECB PATTERN(EXAMPLE)



Communication allowed:

	Entity	Boundary	Control
Entity			X
Boundary			X
Control	X	X	X

ROBUSTNESS DIAGRAM- ECB PATTERN(EXAMPLE)



Here, actors (As Entity classes) can communicate with other entities

Communication allowed:

	Entity	Boundary	Control
Entity	X		X
Boundary			X
Control	X	X	X

That is all