Use Case Realization

**Lecture # 13, 14, 15
04, 06,07 Oct**

Rubab Jaffar
rubab.jaffar@nu.edu.pk

Software Design and Analysis
CS-3004

Today's Outline

- Analysis and Design
- Use case Realization
- Analysis Classes
- ECB Pattern

Analysis and Design

- **Analysis**
  - **Focus on understanding the problem**
  - **Idealized design**
  - **System behavior**
  - **System structure**
  - **Functional requirements**
  - **A small analysis model**

- **Design**
  - **Focus on understanding the solution**
  - **Operations and attributes**
  - **Performance**
  - **Close to real code**
  - **Non-functional requirements**
 - **A large model**

Use Case Realization

- use-case realization describes how a particular use case is realized within the Design Model, in terms of collaborating objects.

  o A use-case realization is one possible realization of a use case.

  o A use-case realization in the Design Model can be traced to a use case in the Use-Case Model.

  o A realization relationship is drawn from the use-case realization to the use case it realizes.

- A use-case realization can be represented using set of diagrams.

Use Case Realization

- 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
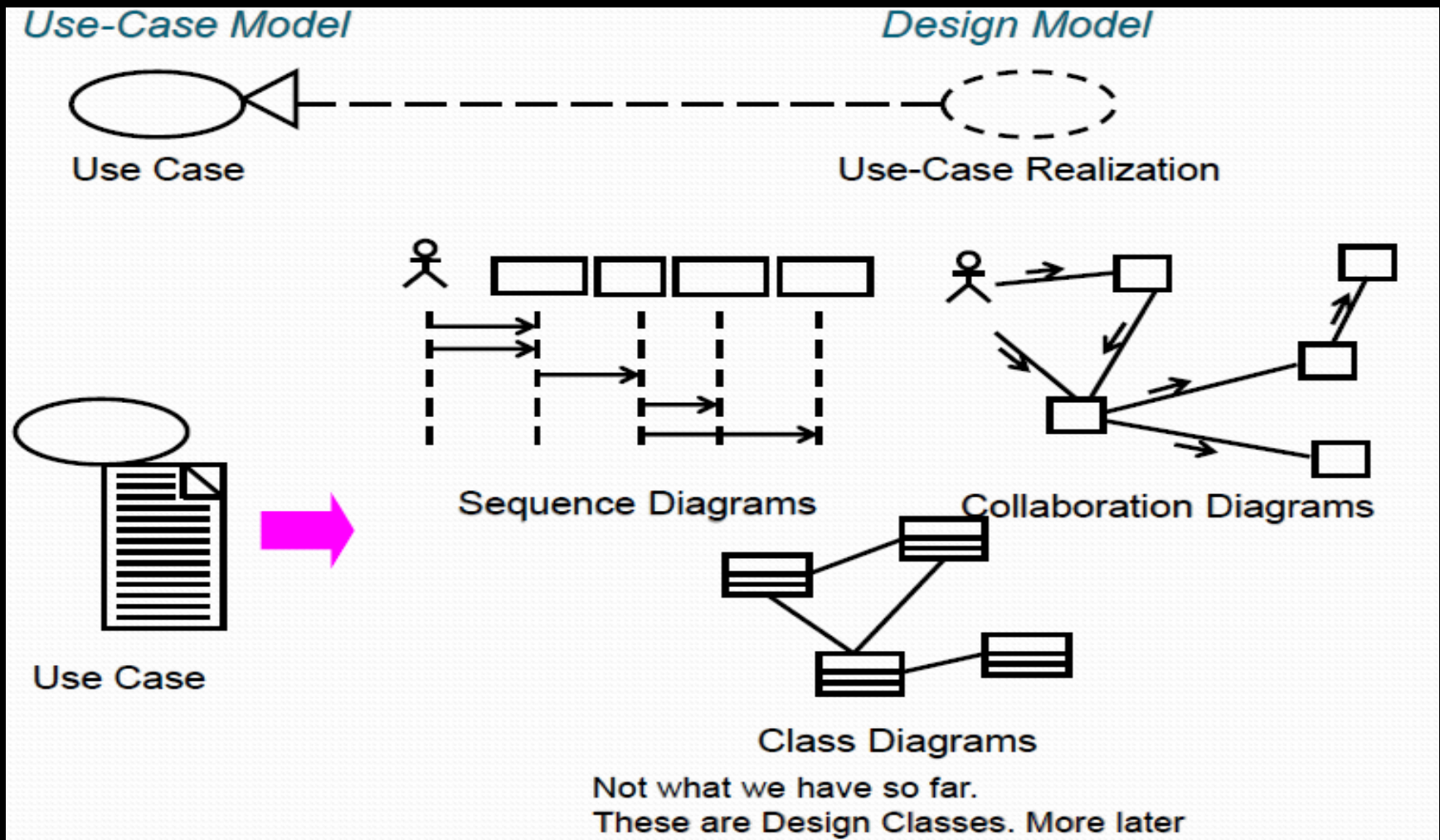
Analysis Classes - Early Conceptual Model

- **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.**

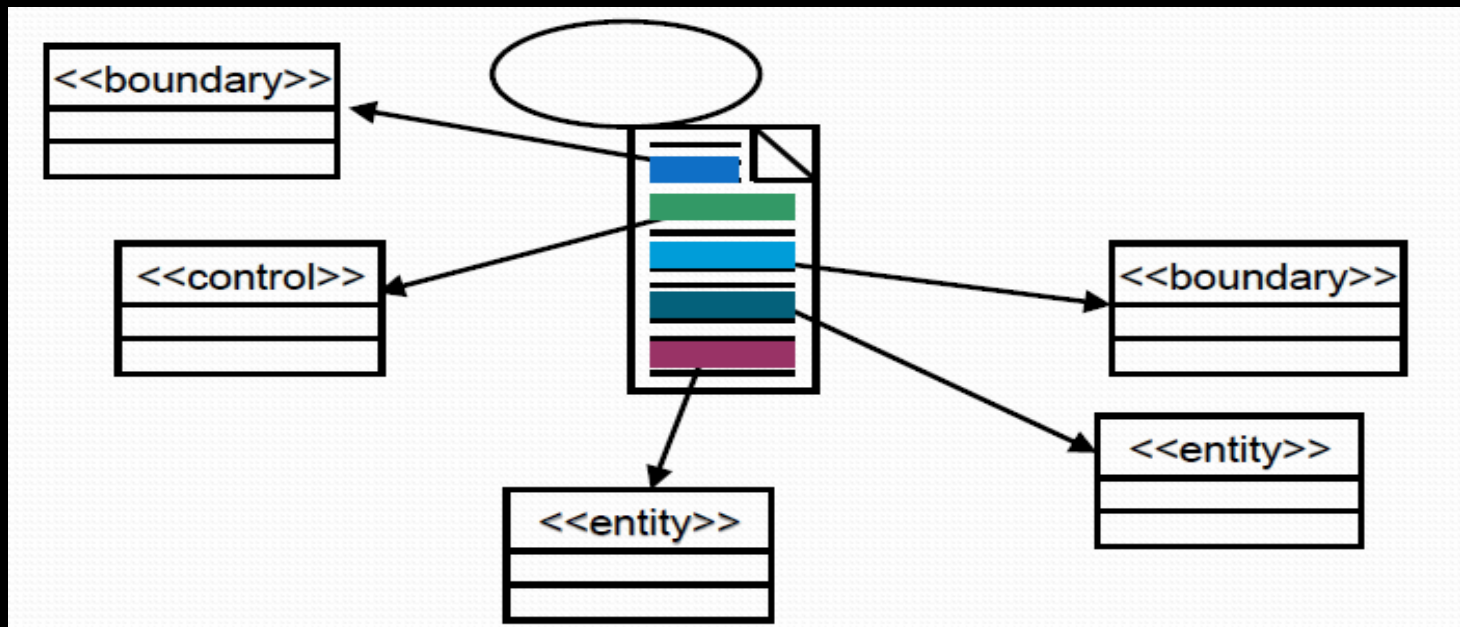# Identifying Candidate Classes from Behavior

- 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.....
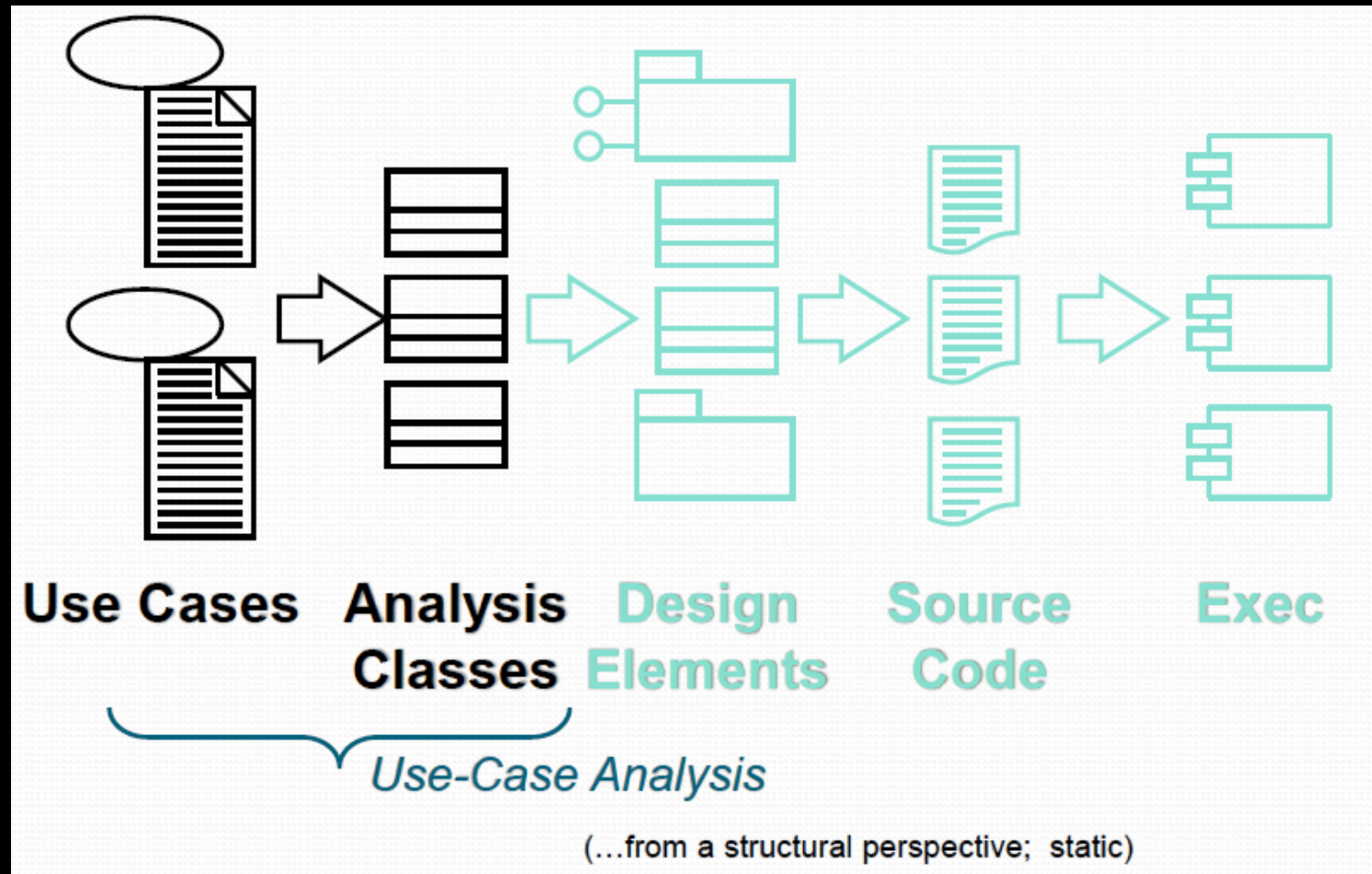
What is an Analysis Class?
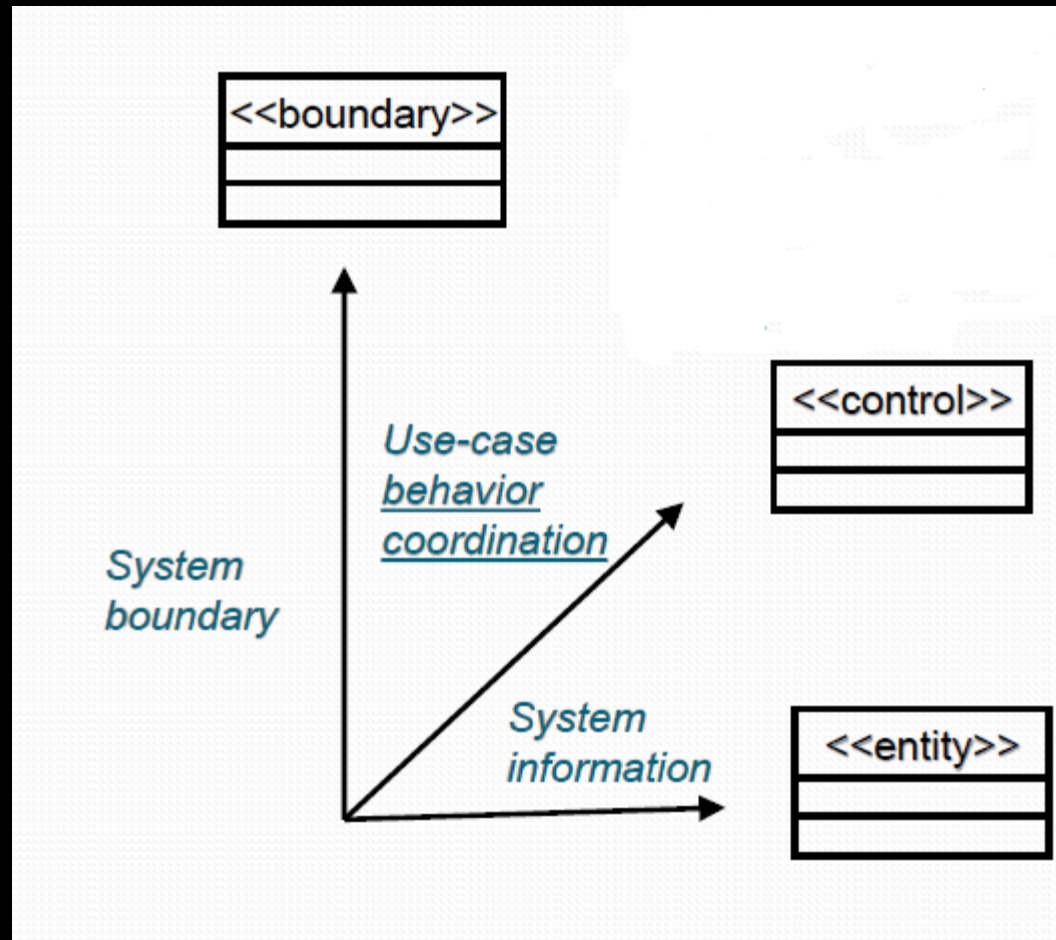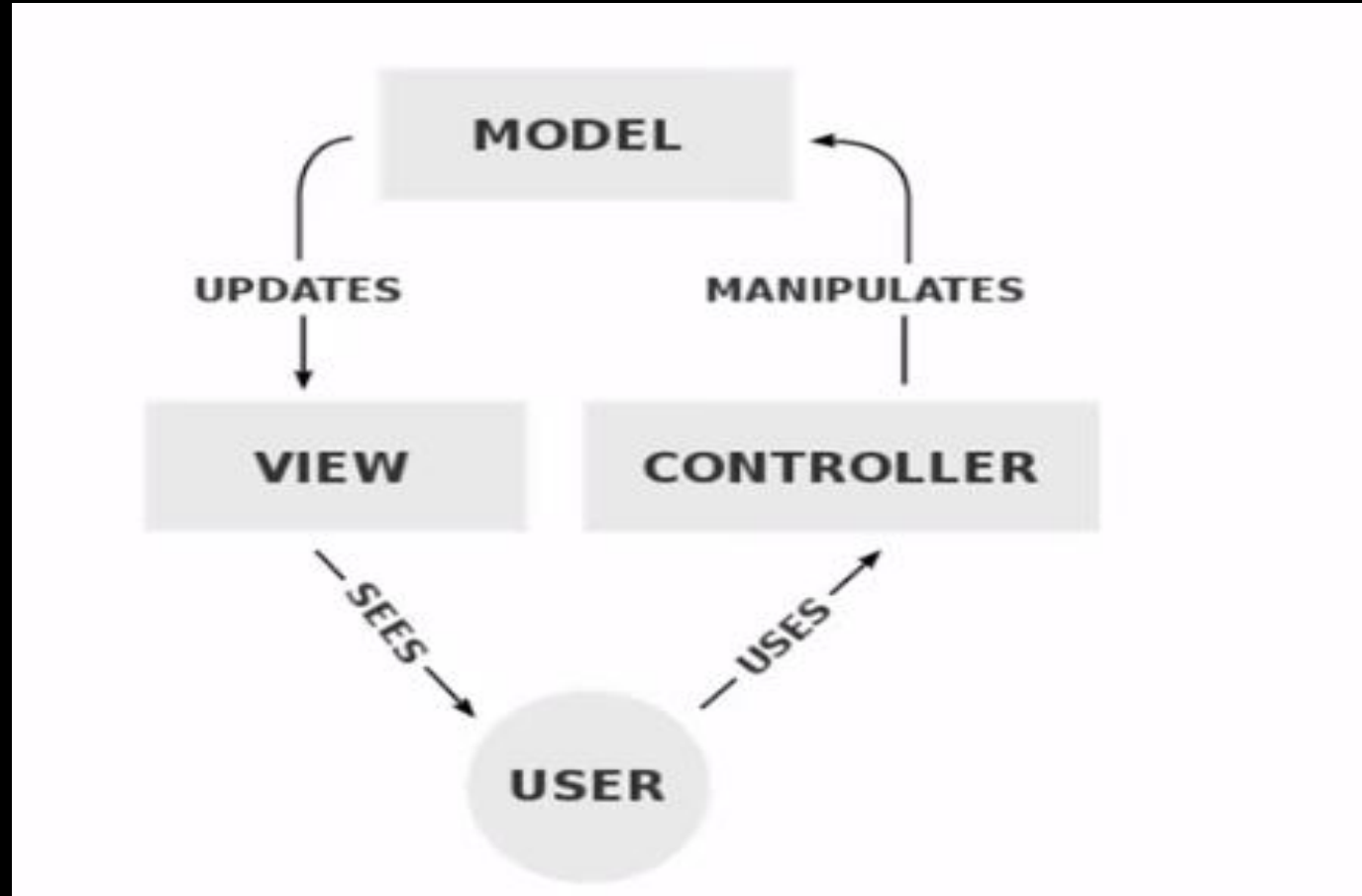
Find Classes From Use-Case Behavior

- **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.**
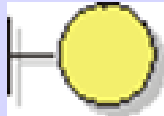
Analysis Classes: A First Step Towards Executables

Use Cases **Analysis Classes** **Design Elements** **Source Code** **Exec**

Use-Case Analysis

(…from a structural perspective; static)

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**

Analysis class stereotype → <<boundary>>
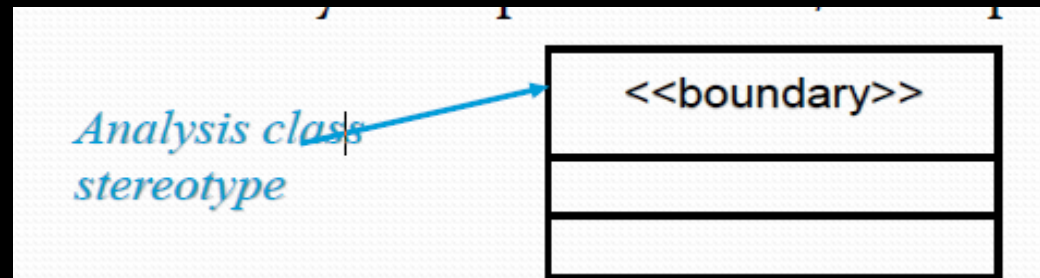
Boundary Class

- **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.
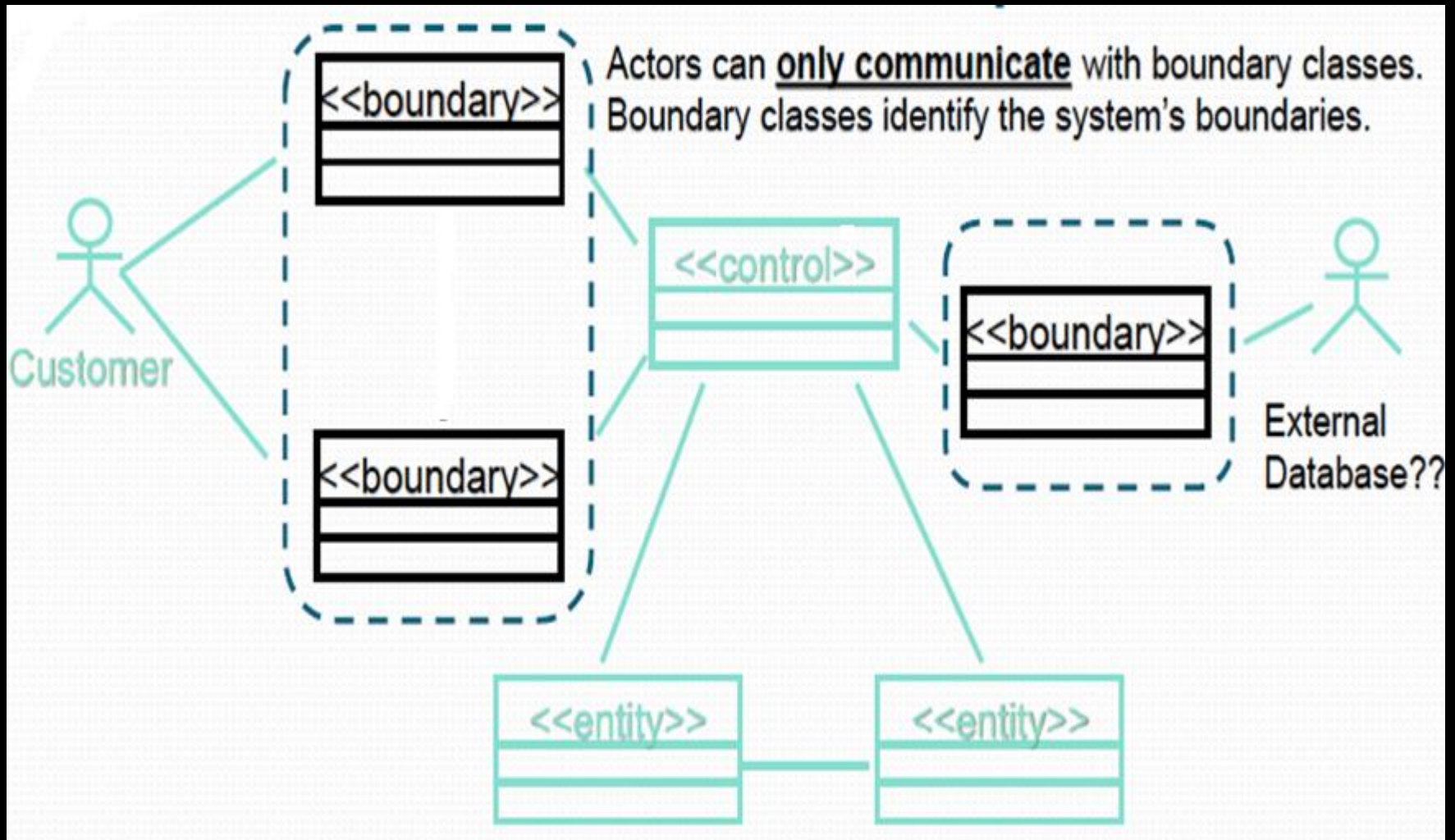
Boundary Class- More

- **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.**

Example: Finding Boundary Class

One boundary class per actor/use case pair:

Student → Register for Courses → Course Catalog System

<<boundary>> RegisterForCoursesForm

<<boundary>> CourseCatalogSystem
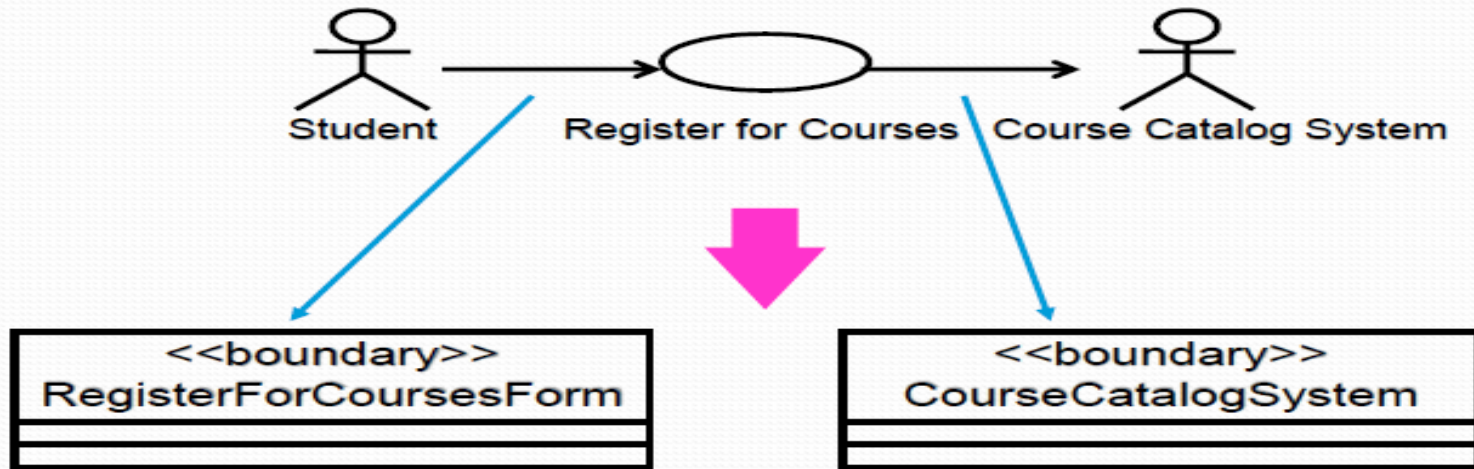
- The RegisterForCoursesForm contains a Student's "schedule-in-progress".
- It displays a list of Course Offerings for the current semester from which the Student may select to be added to his/her Schedule.
- The CourseCatalogSystem interfaces with the legacy system that provides the complete catalog of all courses offered by the university.

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**

Control Classes

- **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**

What is an Entity Class? (recall: boundary, entity, control…)

- Key abstractions of the system

Sources for entity Classes:

Glossary
Use-Case Flow of Events
Business domain model

Glossary

Use Case

Architectural Analysis Abstractions

Business-Domain Model

Analysis class stereotype

<<entity>>

Entity classes show the <u>logical data structure</u>, which will help us understand what the system is supposed to offer to its users.

Environment Independent
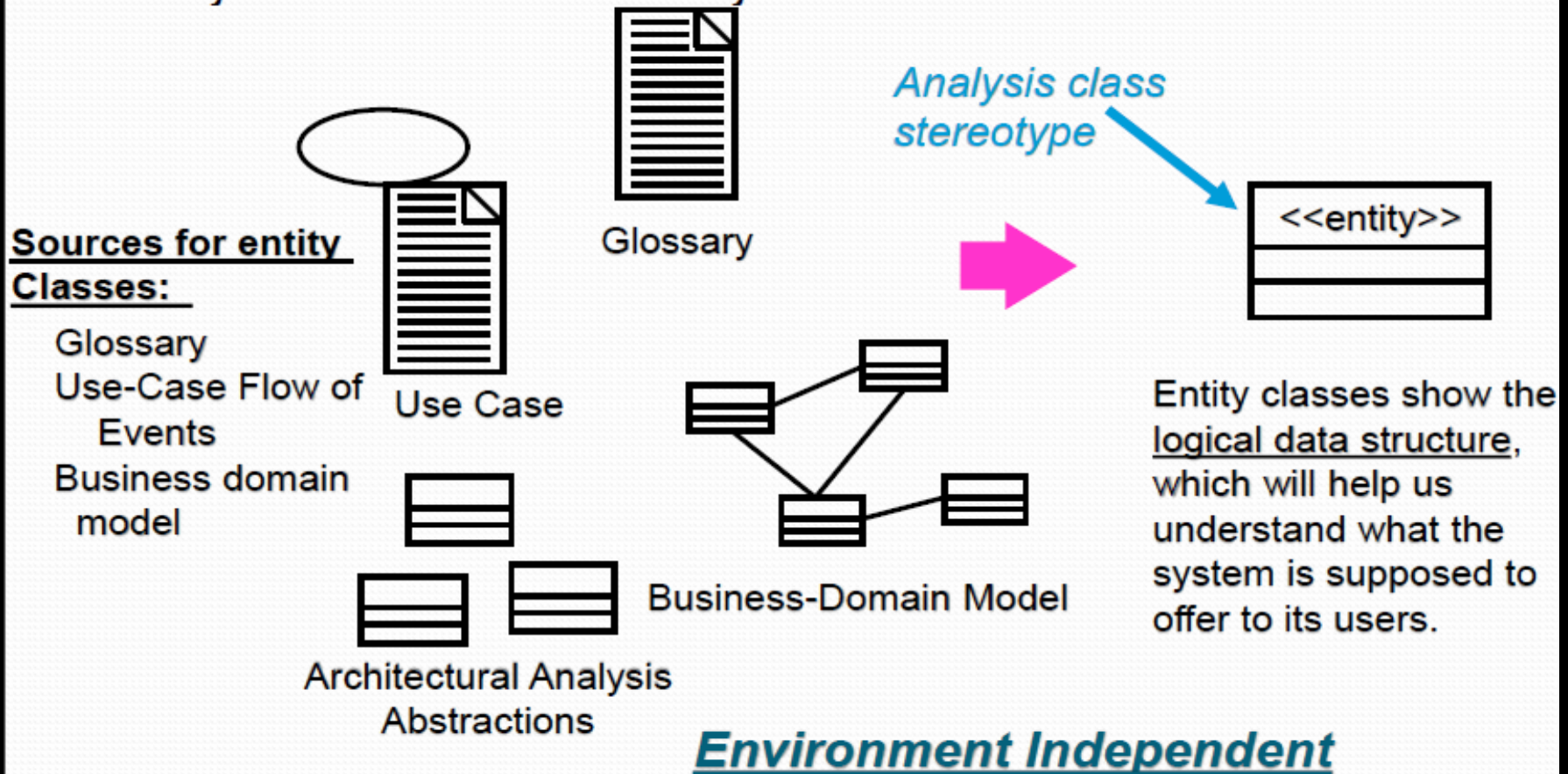
The Role of an Entity Class

Entity objects can have complicated behavior; however, unlike other objects, this behavior is strongly related to the phenomenon the entity object represents.
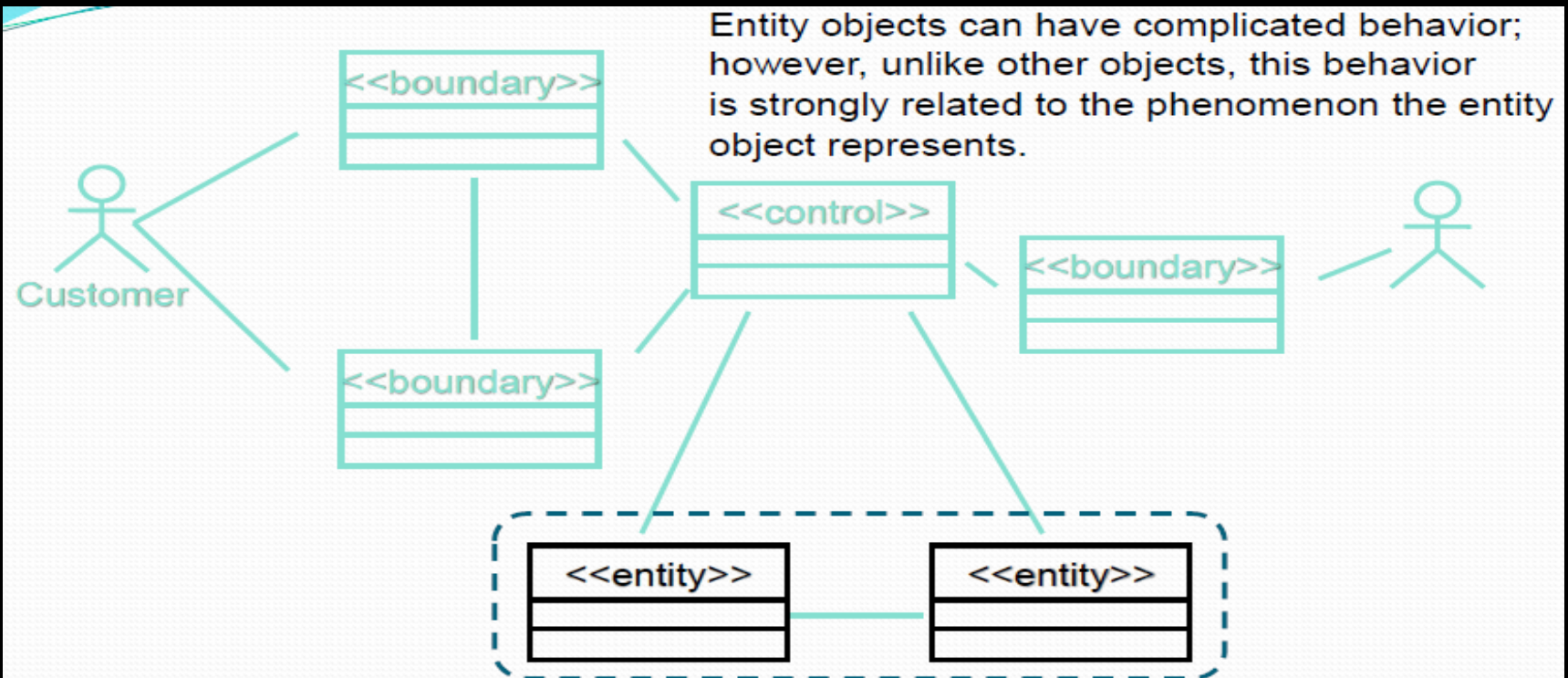
Store and manage information in the system

The values of its attributes and relationships are often given by an actor. Entity objects are **independent** of the environment (actors)

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)
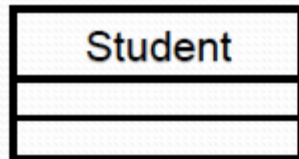
Student

A person enrolled in classes at the university
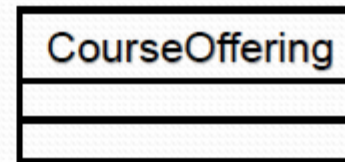
CourseOffering

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

Schedule
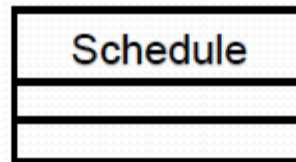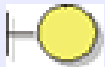
The courses a student has selected for current semester

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
	→	
	←→	
	←→	
	←→	
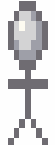
Communication Not Allowed within the ECB
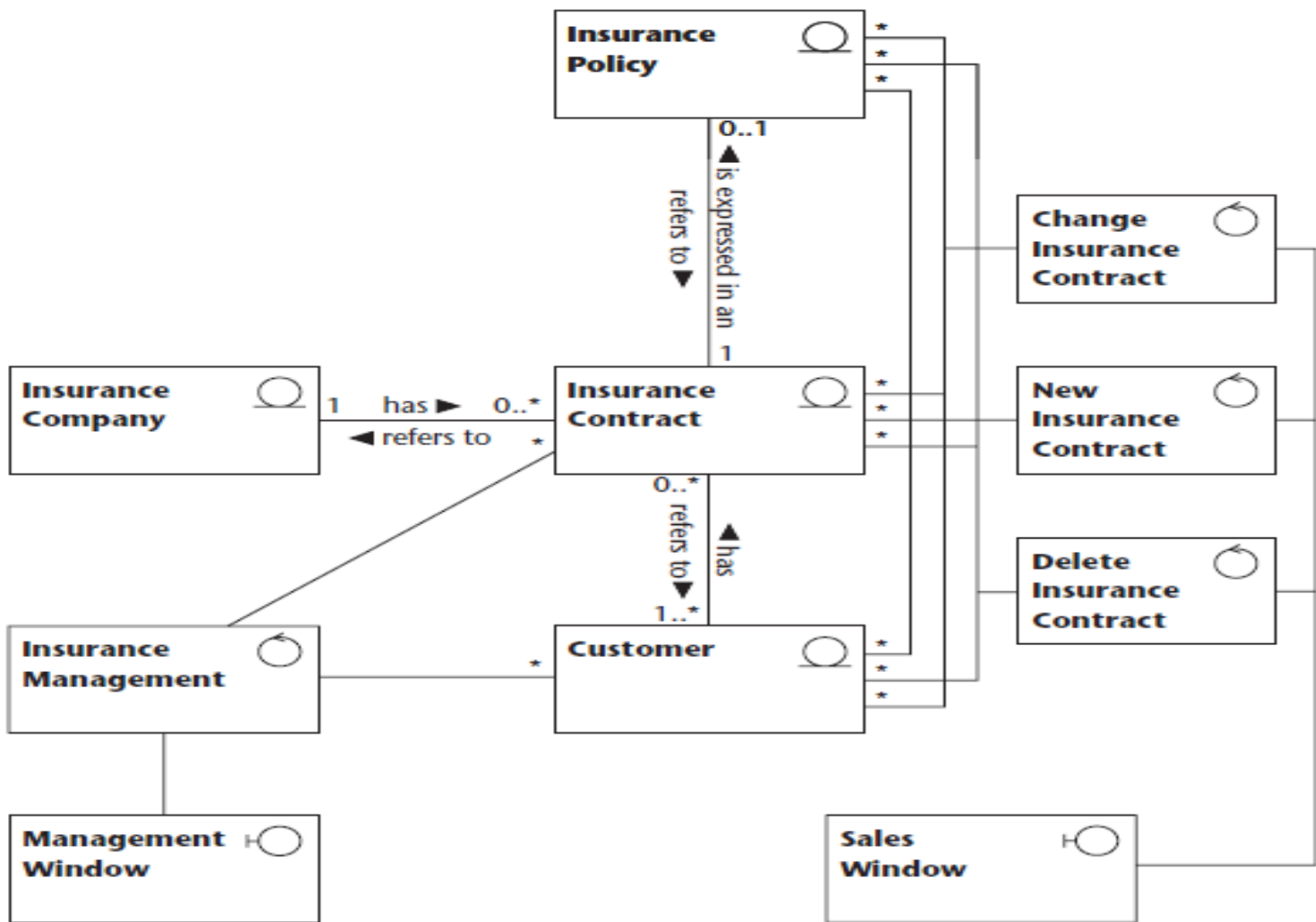
The entity classes capture the core business, in this case, the insurance business. The entity classes are Insurance Policy, Insurance Contract, Customer, and Insurance Company. The control classes Change Insurance Contract, New Insurance Contract, and Delete Insurance Contract serve the boundary class 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 <u>Login Page</u> to the <u>User</u>
🧍 2	<u>User</u> enters username and password
🖥 3	System validates the <u>User</u> details against the <u>Account</u>

Example

- The system displays the Edit shopping cart page.

- The user clicks the checkout button; the system displays the delivery address page showing any addresses already registered for the user's account.

- The user selects the address; the system sets the delivery address in the order.

Example

- From the student detail page, the teacher clicks on the ''Add courses'' button and the system displays the list of courses on a 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 course registration on the confirmation screen.

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.
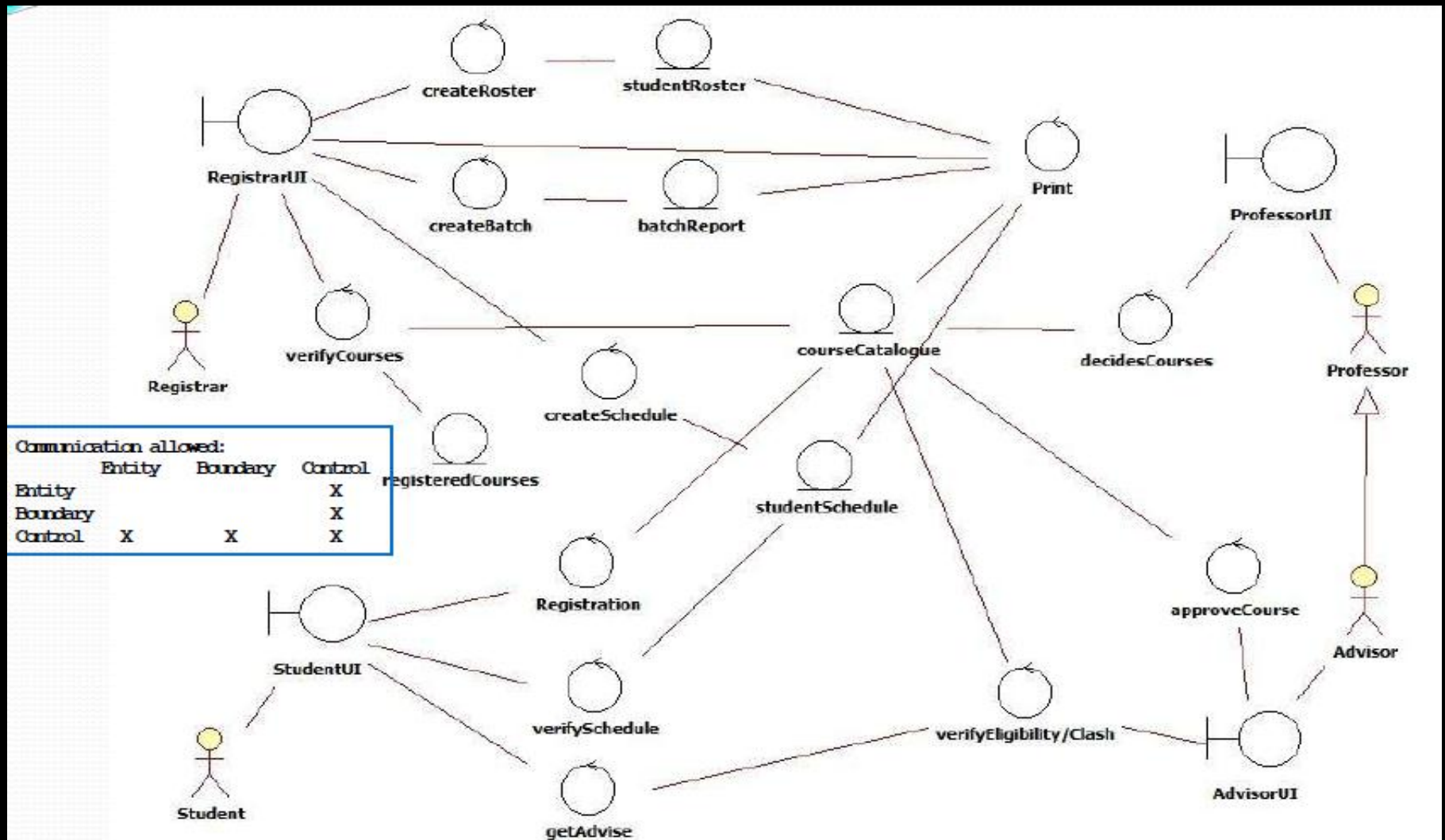
Robustness Diagram- ECB Pattern(Example)

- **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.**
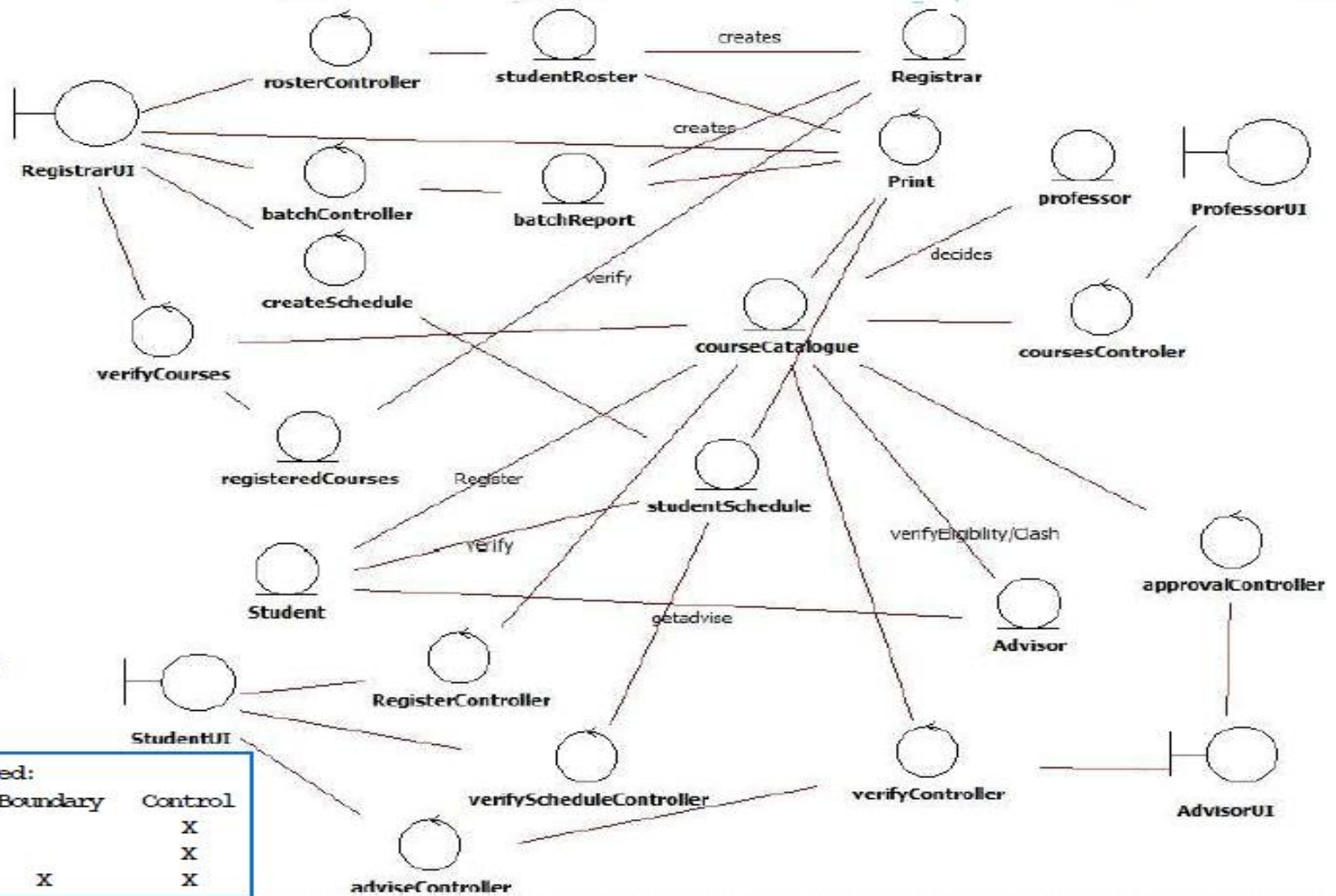
Robustness Diagram- ECB Pattern(Example)

- 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)

Robustness Diagram- ECB Pattern(Example)

That is all