



Entity, Control & Boundary Classes

Lecture 7

Class Diagram

- Class diagram describes the types of objects in the system and the various kinds of relationships that exist among them.
- It also shows the attributes and services of a class and the constraints that apply to the way objects are connected.

Class

Stereotype:
Type of the Class;
entity for Analysis.

Name of the Class



<<entity>>
Patient

Attributes of the
Class



-name : CHAR
-dateOfBirth: DATE

Operations of the
Class



+getName ()

Use Case Realization

- A use-case realization describes how a particular use case is realized within the Design Model, in terms of collaborating objects.
 - A use-case realization is **one possible realization** of a use case.
 - A use-case realization in the Design Model can be ***traced*** to a use case in the Use-Case Model.
- Used to remind us the connection between the requirements expressed as use case and the object design that satisfies the requirement.

Diagramming Representation of Use Case Realization

- **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.
 - 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.

Use Case Model

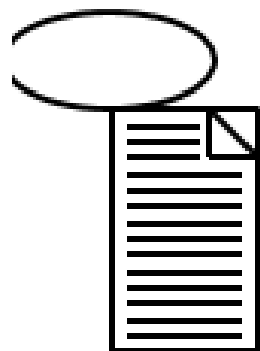


Use Case

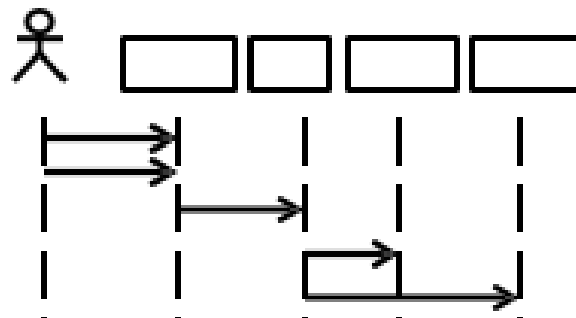
Design Model



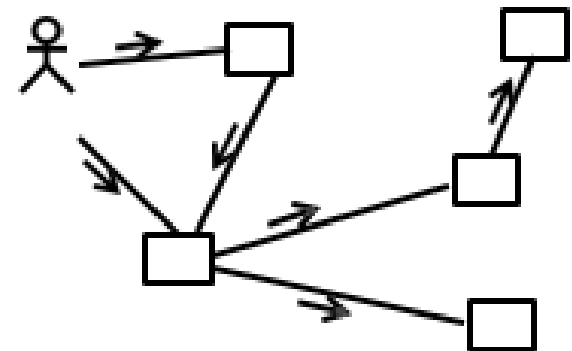
Use-Case Realization



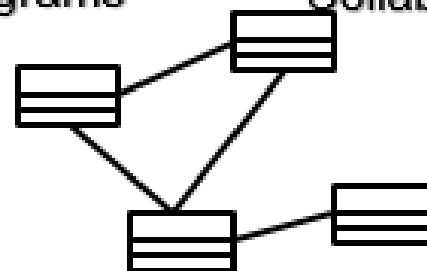
Use Case



Sequence Diagrams



Collaboration Diagrams

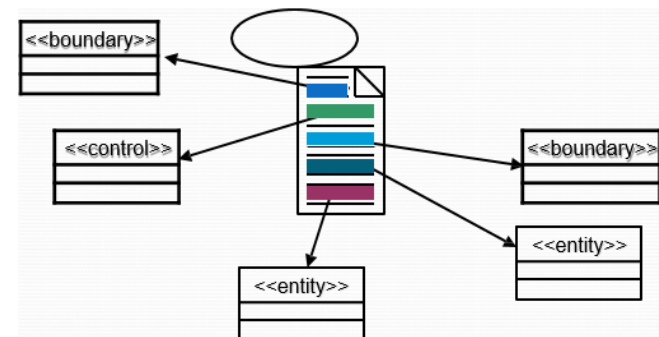


Class Diagrams

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

Identifying Candidate Classes from Behavior

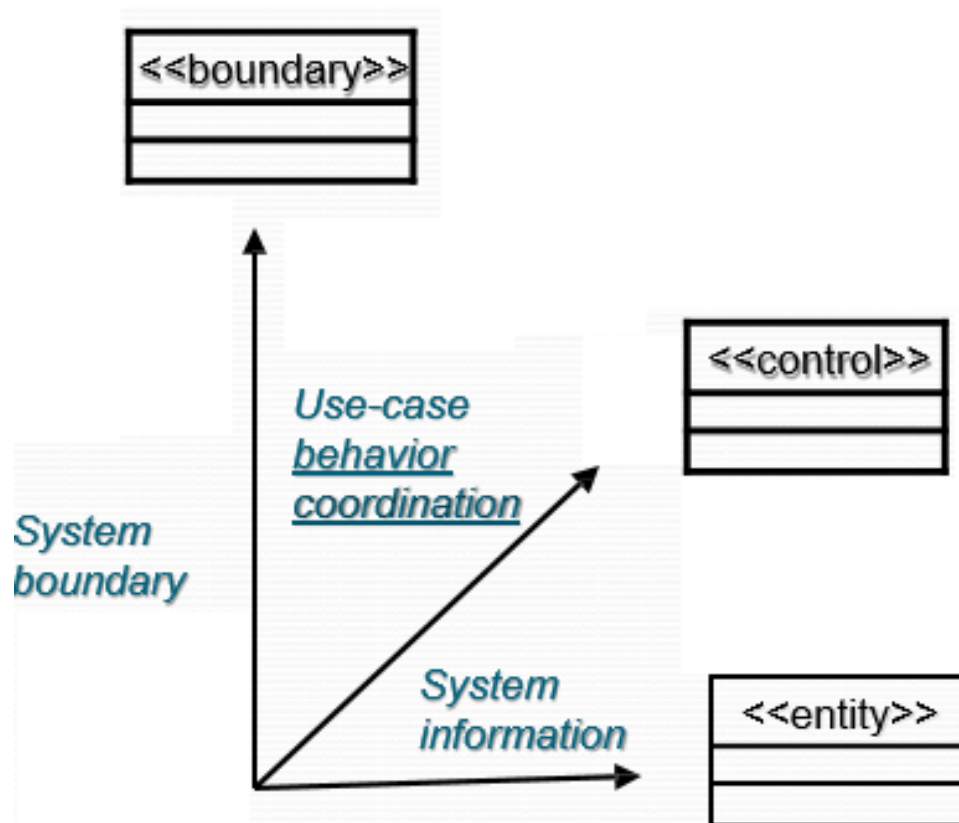
- Will use three perspectives of the system to identify these classes.
 - The '**boundary**' between the system and its actors
 - The '**entity**' the system uses
 - The '**control logic**' of the system
- Will use stereotypes to represent these perspectives (**boundary**, **control**, **entity**)
 - 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.



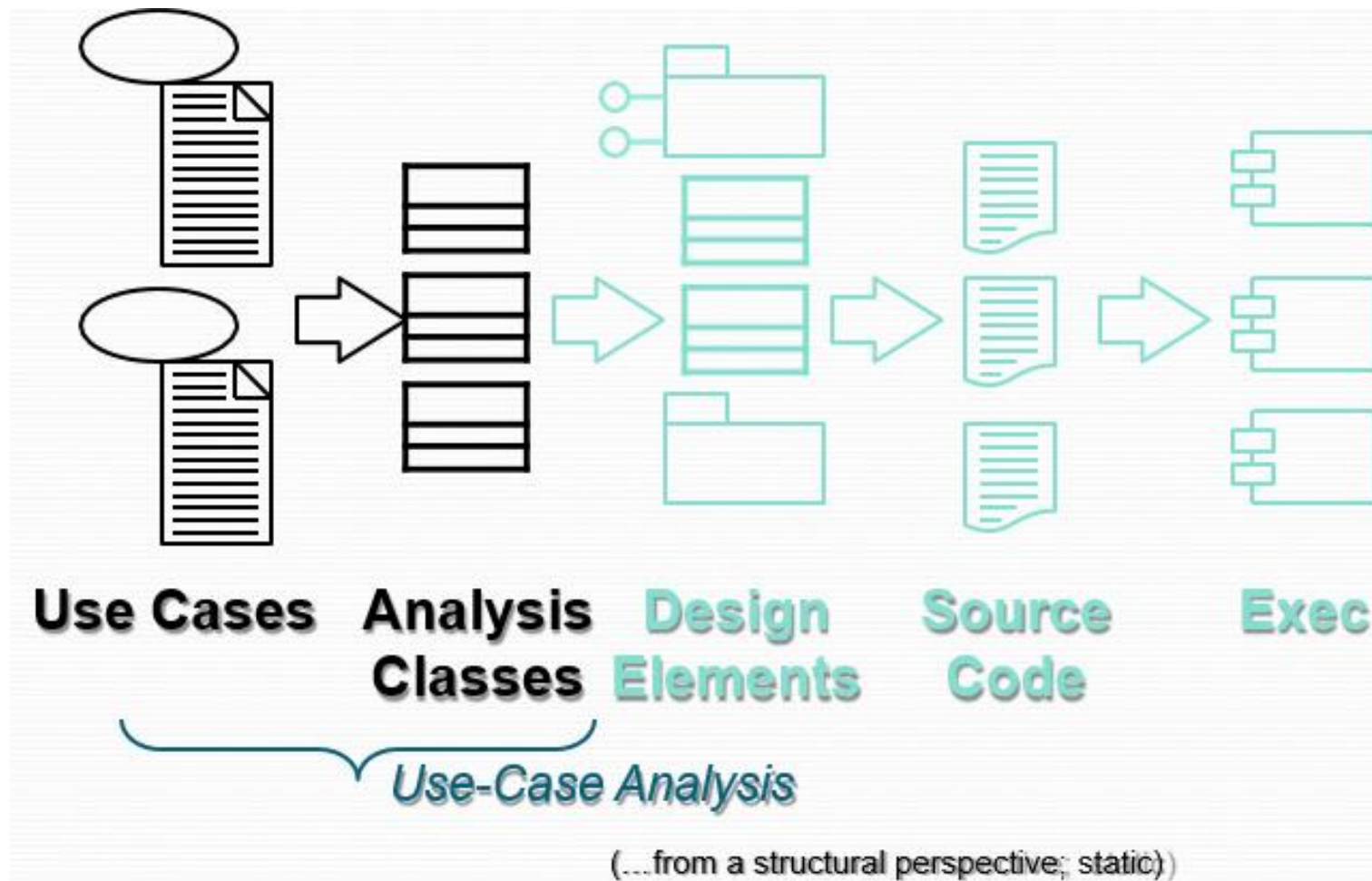
What is an Analysis Class?

Can use with the name of the stereotype
In guillemets or as symbols with unique icons.

Finding a candidate set of classes is the first part of transforming a mere statement of required behavior to a description of how the System will work.



Analysis Classes: First Step towards Executables



Application Class Stereotypes

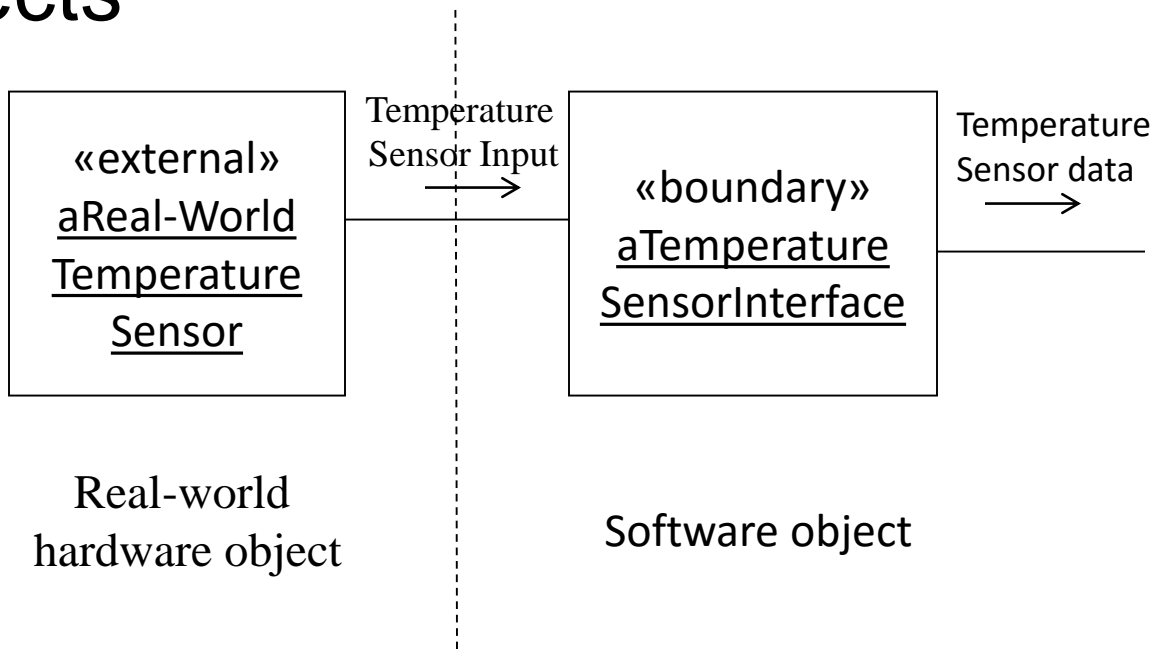
- «boundary»
 - Provides the interface for external interactions
 - 1:1 correlation with «external» classes
- «entity»
 - Manages persistent data
- «control»
 - Central controller for one or more use case scenarios

Boundary Classes

- Provide the interface to a user or another system.
- Handles communication between system surroundings and the inside of the system
 - User interface classes
 - System interface classes
 - Device interface classes

Identifying «boundary» Classes

- One-to-one mapping with «external» objects



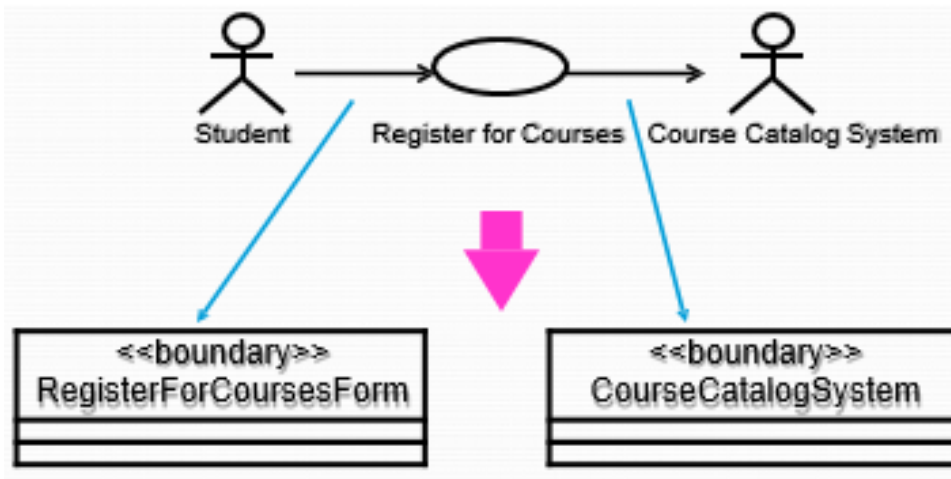
Hardware/software boundary
(for illustration only – not UML)

Boundary Class

- 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 Classes

- One boundary class per actor/use case pair:



- 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.

Boundary Class (Interface)

- User Interface Classes
 - Concentrate on what information is presented to the user
 - Do NOT concentrate on the UI details
 - Analysis Classes are meant to be a first cut at the abstraction of the system.
 - The boundary classes may be used as ‘holding places’ for GUI classes.
 - Do not do a GUI design in analysis, but isolate all environment-dependent behavior. (Likely you may be able to reverse engineer a GUI component and tailor it.)
 - The expansion, refinement and replacement of these boundary classes with actual user interface classes is a very important activity of Class Design.
 - If prototyping the interface has been done, these screen dumps or sketches may be associated with a boundary class.
 - Only model the key abstractions of the system – not every button, list, etc. in the GUI.

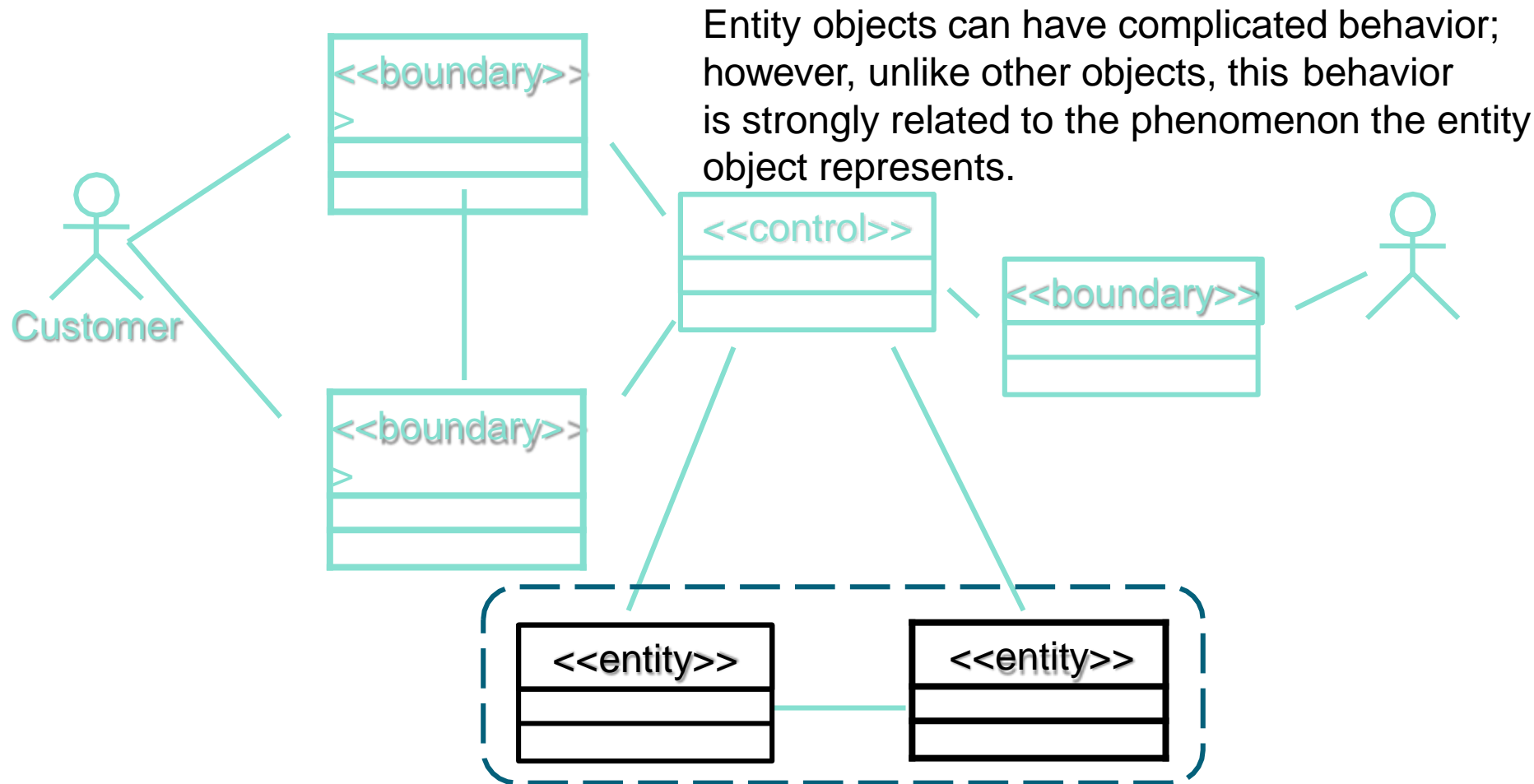
Boundary Class (System & Device)

- System and Device Interface Classes
 - Concentrate on what protocols must be defined
 - Note that your application must interface with an existing 'information source.'
 - Do NOT concentrate on how the protocols will be implemented
- If the interface to an existing system or device is already well-defined, the boundary class responsibilities should be derived directly from the interface definition.
- If there is a working communication with the external system or device, make note of it for later reference during design.

Entity Classes

- Fundamental building block which perform internal tasks
- Represent real world entity
- 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.

The Role of an Entity Class



Store and manage information in the system

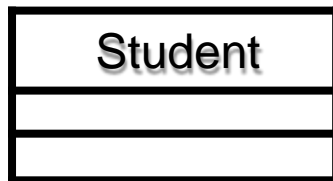
The values of its attributes and relationships are often given by an actor. Entity objects are **independent** of the environment (actors)

Candidate Entity Classes

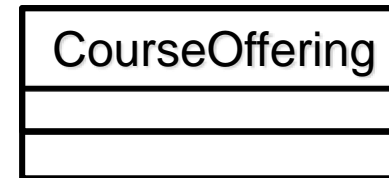
- Sometimes there is a need to model information about an actor within the system. This is not the same as modeling the actor (actors are external by definition).
- For example, a course registration system maintains information about the student which is independent of the fact that the student also 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

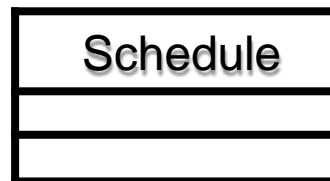
Register for Courses (Create Schedule)



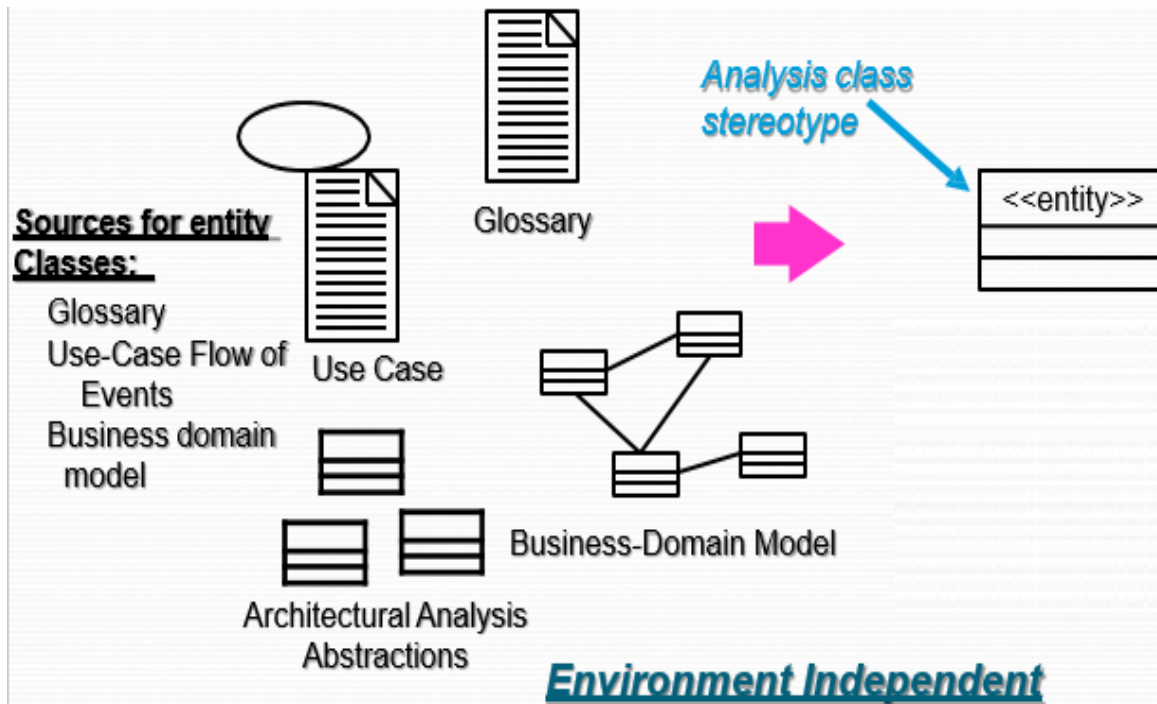
A person enrolled in classes at the university



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



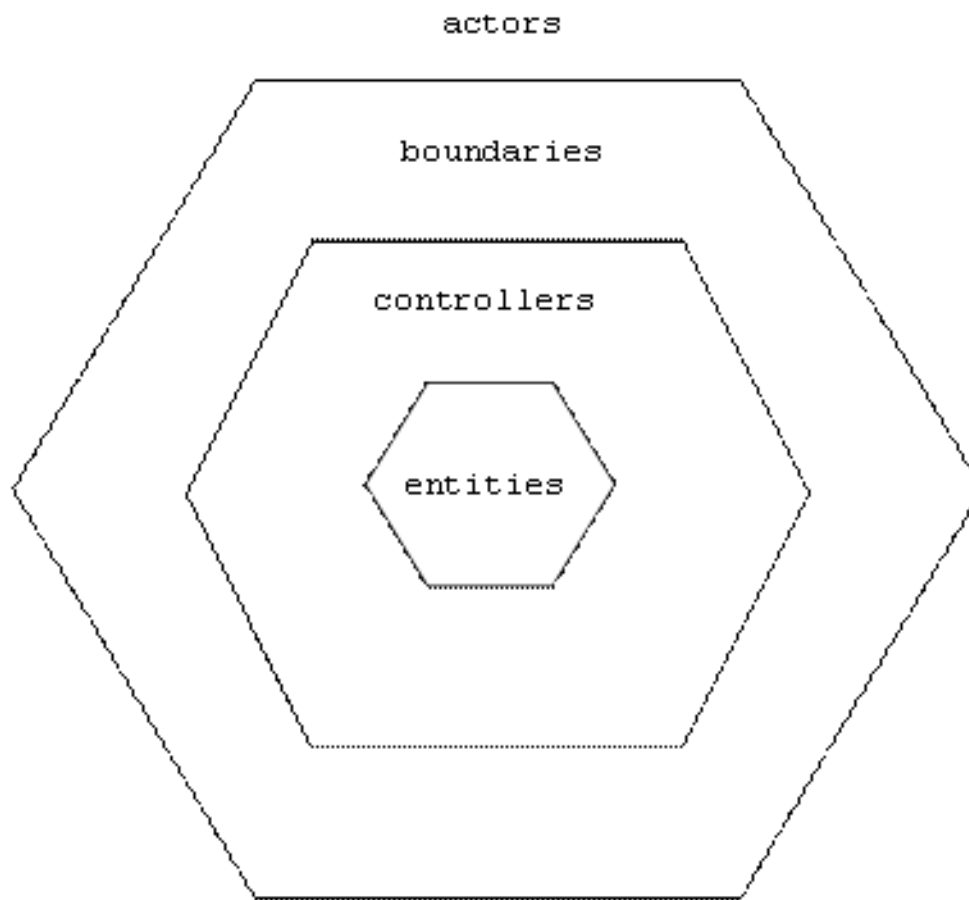
The courses a student has selected for current semester



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

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



Stereotypes and Classes

A Stereotype is a mechanism use to categorize classes.

- Primary class stereotypes in UML

 *Boundary*

 *Entity*

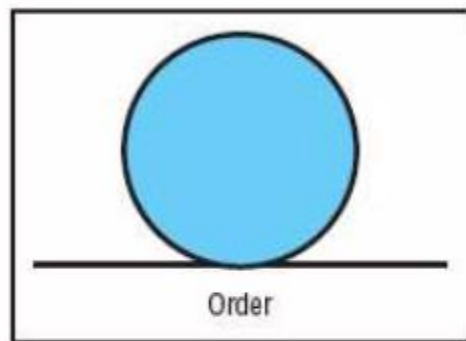
 *Control*

Analysis Classes

- 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

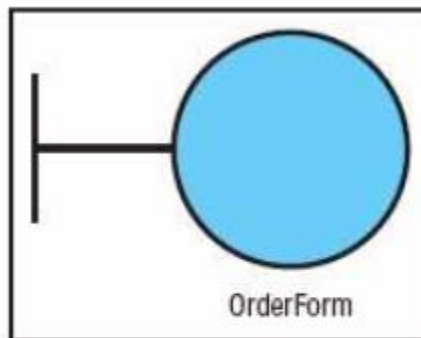
Stereotypes of Analysis Classes

Figure 9.3a
Entity Class Order



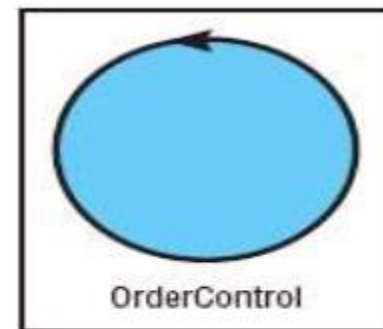
Mostly
corresponds to
conceptual data
model classes

Figure 9.3b
Boundary Class
OrderForm



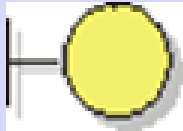


Encapsulates
connections
between actors
and use cases

Figure 9.3c
Control Class
OrderControl















Mostly performs
behaviors
associated with
inner workings of
use cases

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»	

Allowed Communication within the ECB Design Pattern

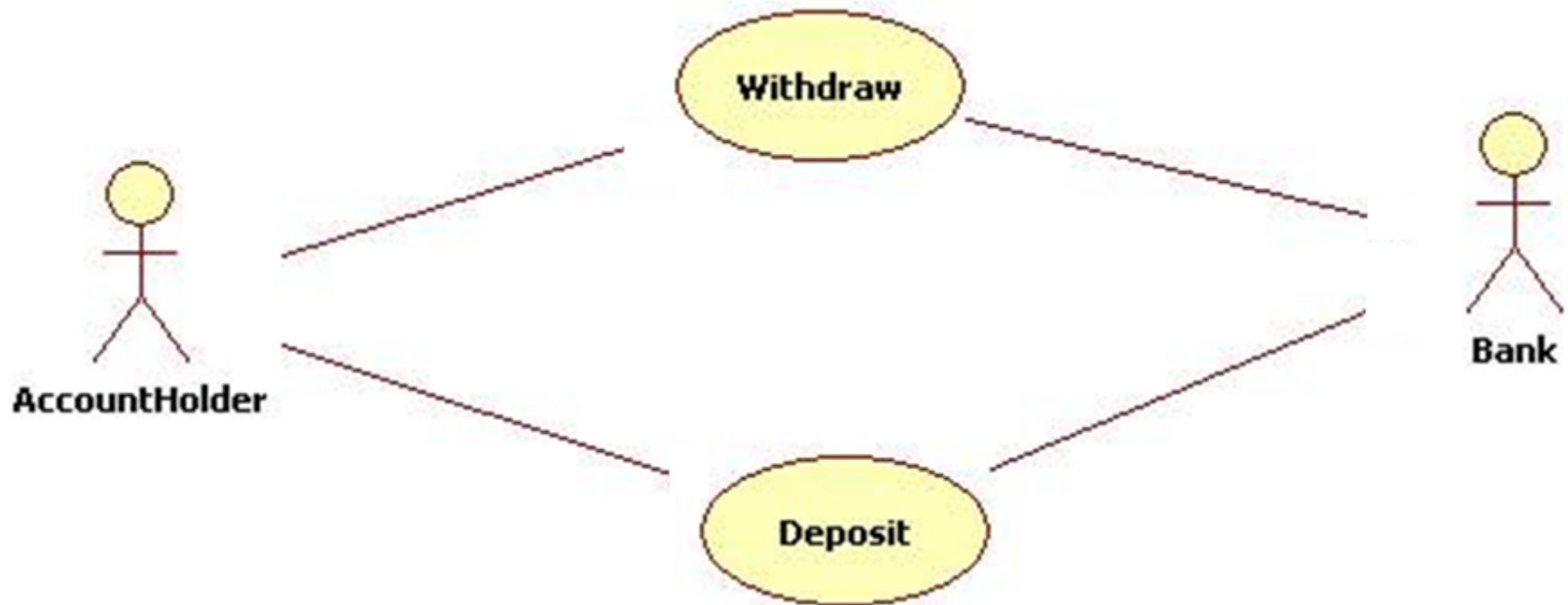
Actor or Object Initiating Communication	Flow of Communication	Target Object
		
		
		
		

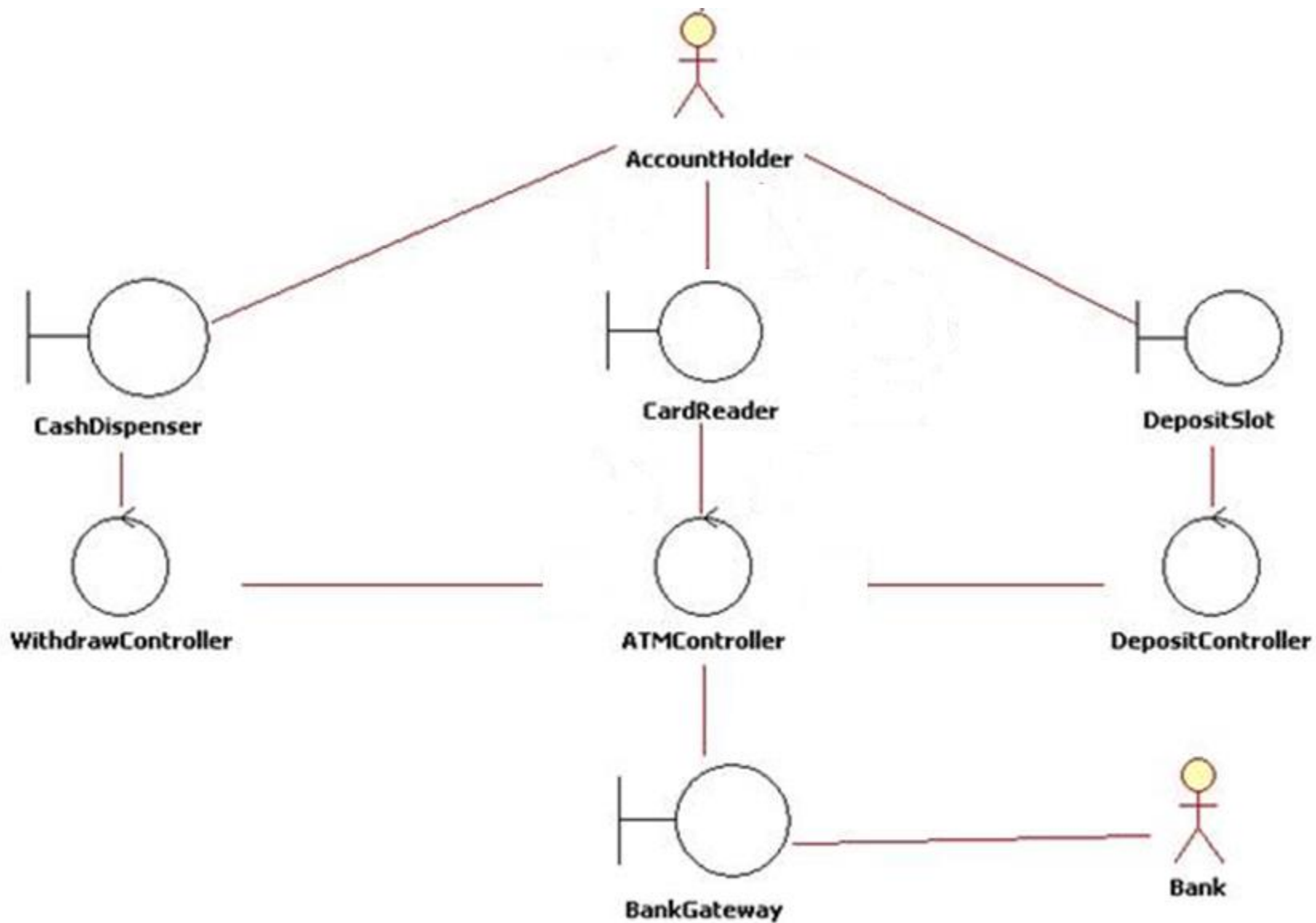
What Is Robustness Analysis?

- 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

ATM Scenario

- The Simple ATM allows account holders to make deposits to and withdraw funds from any accounts held at any branch of the Bank of Pakistan.





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.

Championship Management

- Design a system for organizing championships of table games (chess, go, backgammon, etc.)
 - Requirements:
 - A player should register and log in to the system before using it.
 - Each registered player may announce a championship.
 - Each player is allowed to organize a single championship at a time.
 - Players may join (enter) a championship on a web page – When the sufficient number of participants are present, the organizer starts the championship.
 - After starting a championship, the system must automatically create the pairings in a round-robin system.

User Management (Design the ECB Diagram for the Use case given below)

