University of Dublin
Trinity College

Information Modeling
Using
The Unified Modelling Language
(UML)

… the **art of communication** of the design of information..

Modelling Constraints in UML
Object Constraint Language

Constraints: Motivation

Constraints on UML model elements: **conditions/criteria** that must be true about some aspect of the system

Constraints will make the analysis and design more precise and rigorous

Complement the UML graphical notation and can be useful to use with ALL model elements (e.g. classes, attributes, methods, transitions)

Helps with verification and validation of models

Helps with communication of intent of some aspect of model

Example: What type of constraints in Class Models?

- A class model can define the structure of data
  - *"A payment must include a payer and a recipient"*

- But OCL is needed to define interdependencies between the data
  - *"The payer and the recipient cannot be the same"*
  - *payer.name <> recipient.name*

Expression: Context

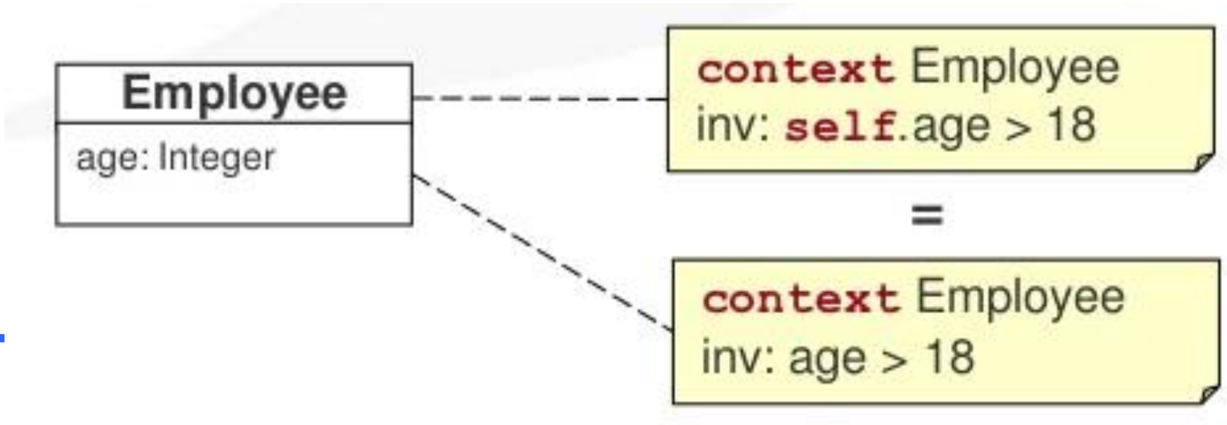
Every OCL constraint has a context, the element that is being constrained (operation, class)

A constraint can be written in a textual form (data dictionary) or attached to model elements as a note

Keyword **`context`** in bold type

The keyword **`self`** in the textual form of the constraint simply refers to the instance of the context class (not always needed but aids readability)

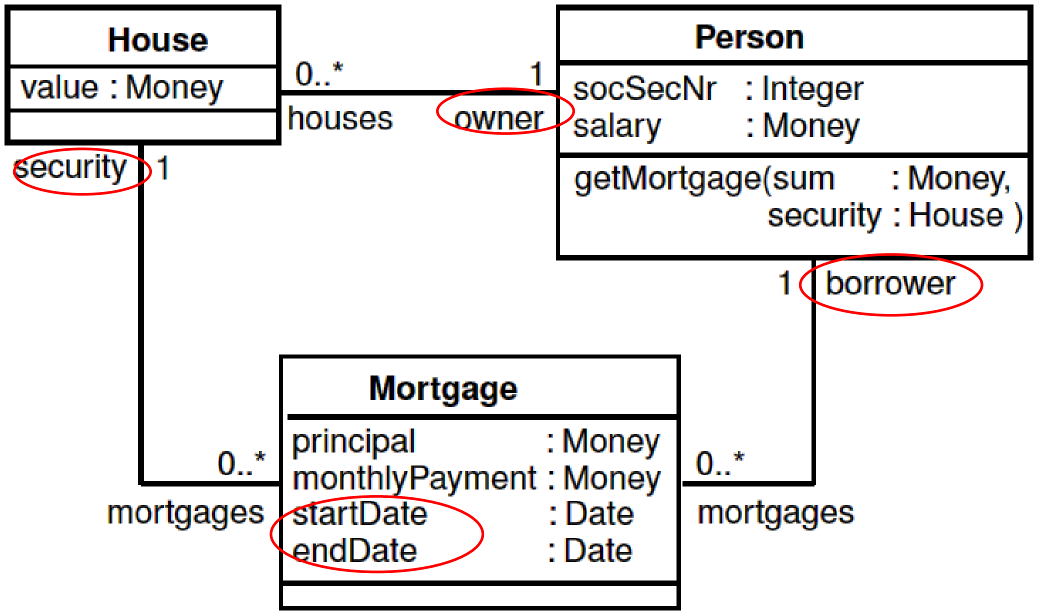
Invariant - Constraint that applies to ALL instances of class (or type or interface) - An expression that evaluates to true if the condition is met.



Example: A Mortgage System

Might want to express the following:

1. A person may have a mortgage only on a house he/she owns.
2. The start date of a mortgage is before its end date.



1. **context** *Mortgage*
   invariant: *self.security.owner = self.borrower*

context *Mortgage*
invariant: *security.owner = borrower*

2. **context** *Mortgage*
   invariant: *self.startDate < self.endDate*

context *Mortgage*
invariant: *startDate < endDate*

UML views and diagrams

Major View	Sub-view	Diagram	Concepts
structural	static	class diagram	association, class, dependency, generalization, interface, realization
	design	internal structure	connector, interface, part, port, provided interface, role, required interface
		collaboration diagram	connector, collaboration, collaboration use, role
		component diagram	component, dependency, port, provided interface, realization, required interface, subsystem
	use case	use case diagram	actor, association, extend, include, use case, generalization

UML views and diagrams cont.

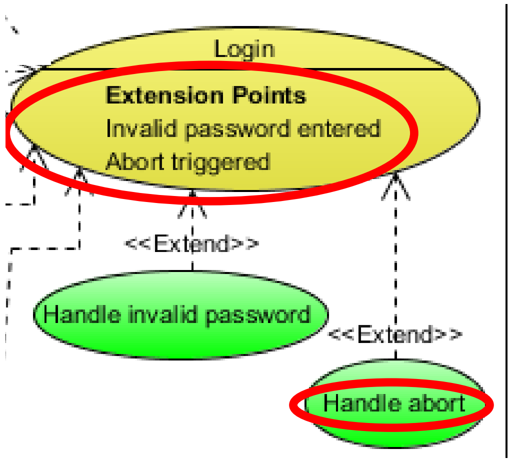
Major View	**Sub-view**	**Diagram**	**Concepts**
dynamic	state machine	state machine diagram	completion transition, do activity, effect, event, region, state, transition, trigger
	activity	activity diagram	action, activity, control flow, control node, data flow, exception, expansion region, fork, join, object node, pin
	interaction	sequence diagram	occurrence specification, execution specification, interaction, lifeline, message, signal
		communication diagram	collaboration, guard condition, message, role, sequence number

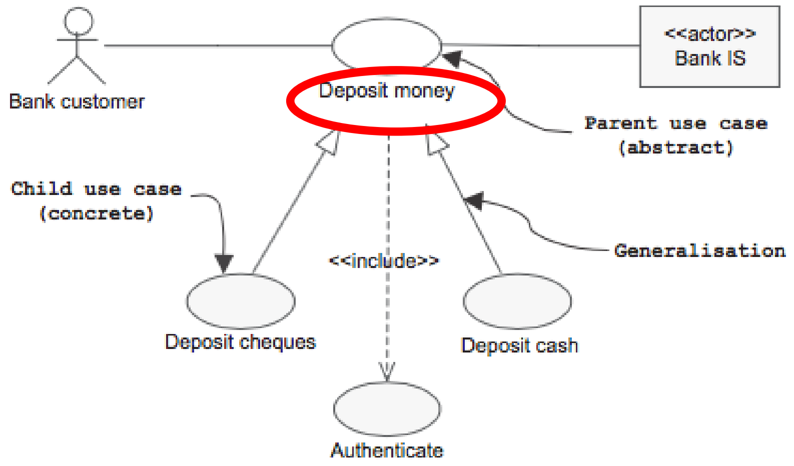
Notation Variations

- For Use Cases

Some Notation Variations for Use Cases

3 Differences to note In These Use Cases





1. Name Under Use Case Bubble
2. Name In Use Case Bubble
3. Use Case Bubble with Info about Extends relationships

UML Notation Variations for Actors

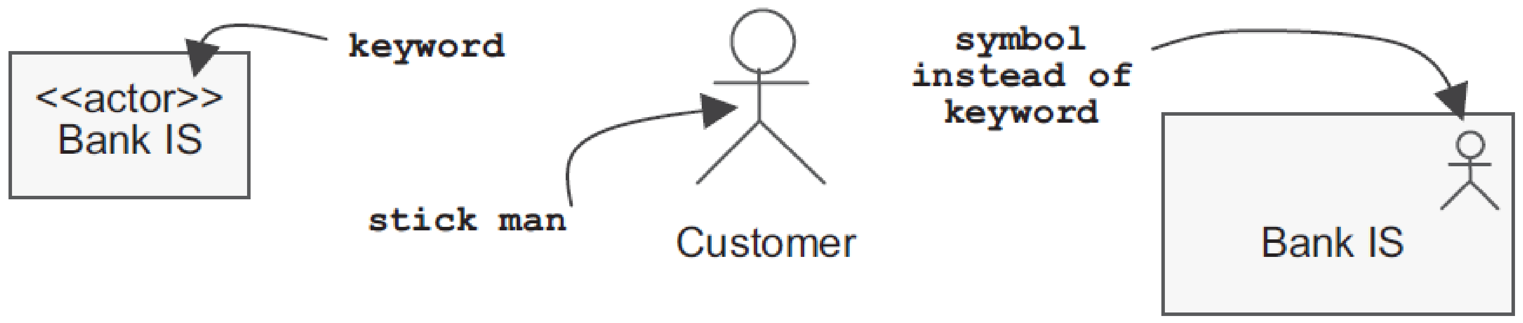
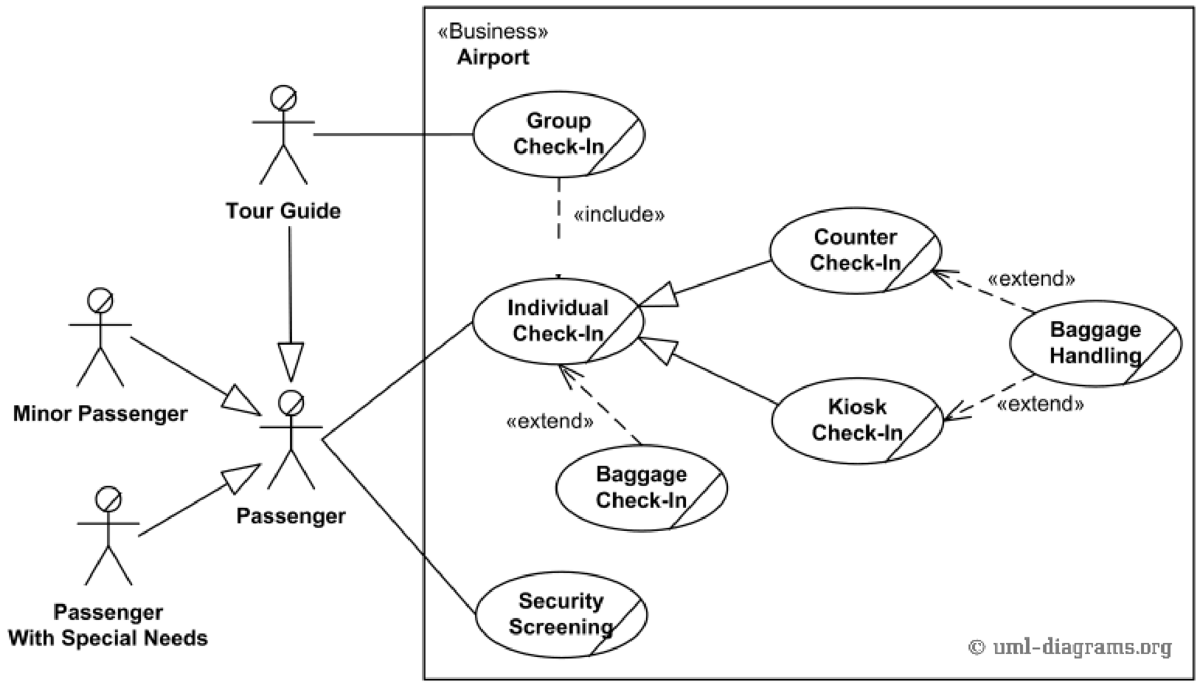


Figure 1.1 Possible graphical representations of an actor

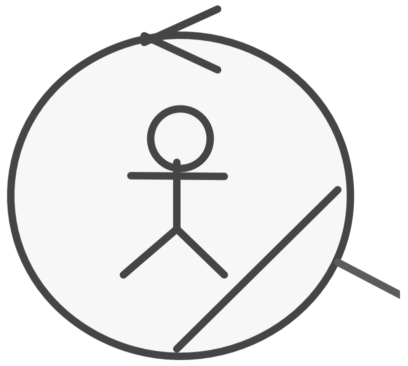
Some Notation Variations for Use Cases
Business Use Cases-

Note the stroke in the right hand corner of use case bubbles and actors heads



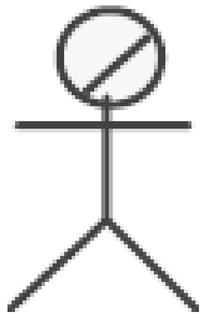
https://www.uml-diagrams.org/airport-checkin-uml-use-case-diagram-example.html

Representing Actors interactions within and outside organisation



Employee

Class representing interaction with Actor **within** Organisation



Training body

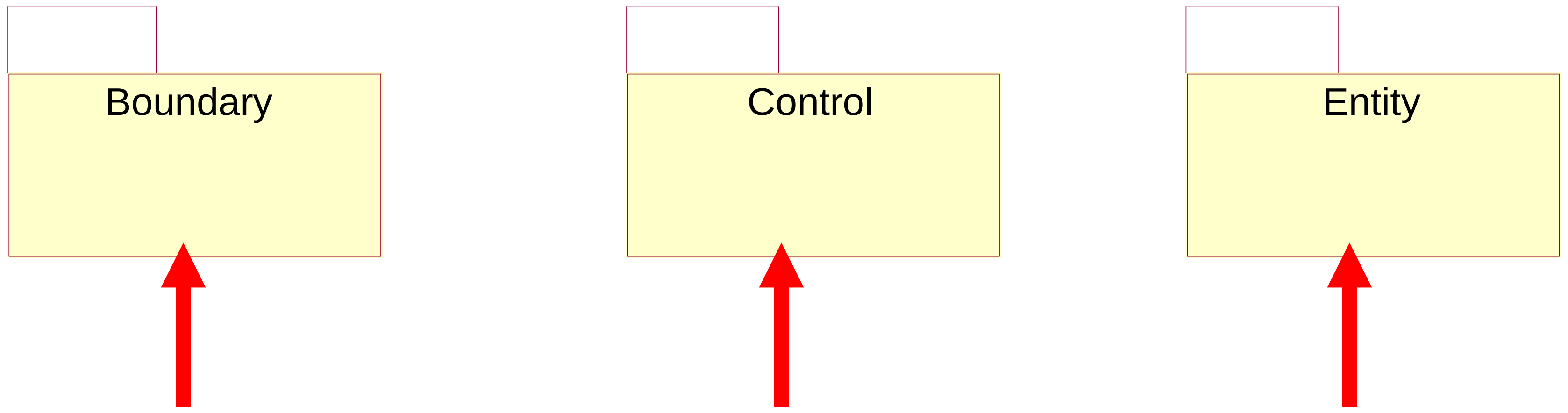
Actor **Outside** Organisation

Notation Variations

- For Class Diagrams

Boundary/Control/Entity Approach to Class diagrams

The three categories Align with Three Tier Computing thinking: Client, Server, Database



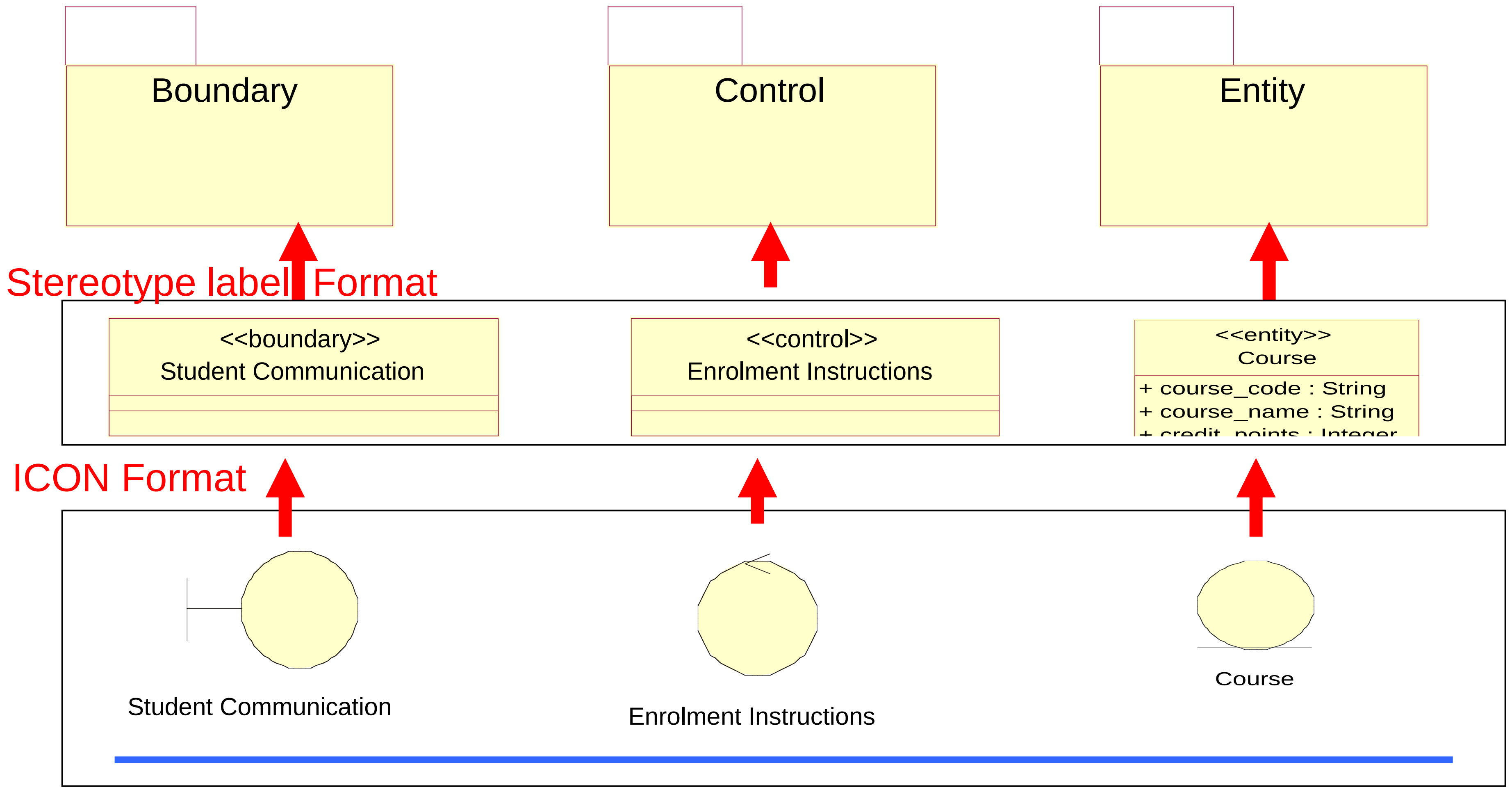
Contain classes that represent an interface between an actor and the system. Often persist beyond single session

Contain classes that intercept user/system input events and control execution. Frequently do not persist beyond a session execution

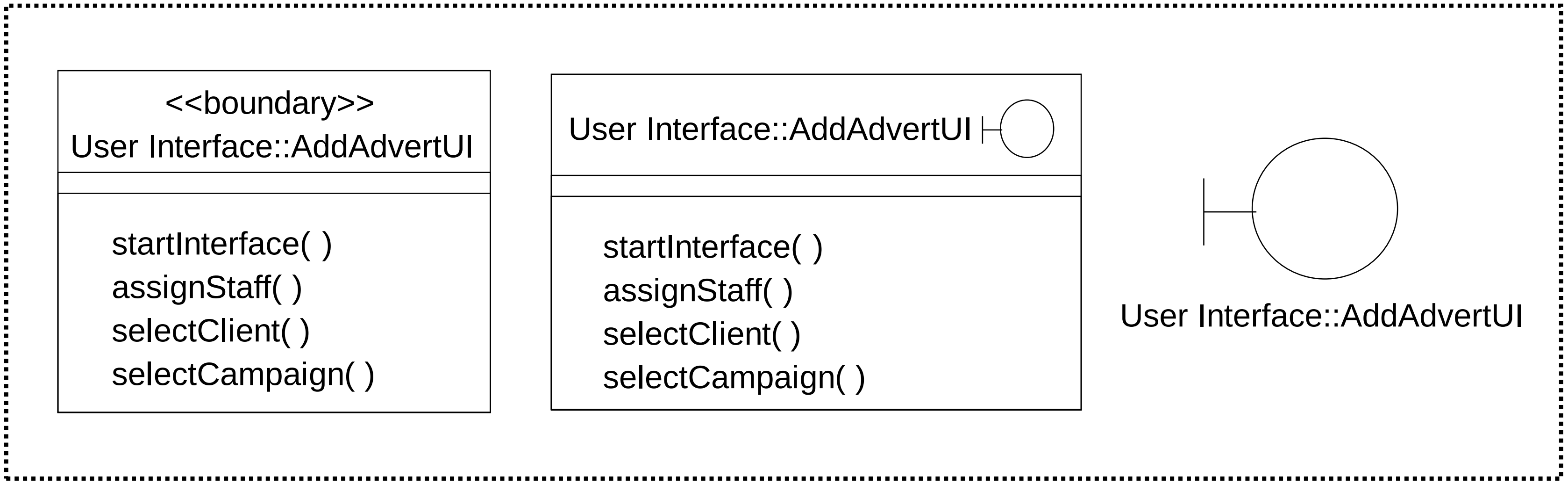
Contain classes that represent entities about which you want to keep information beyond a single session

Boundary/Control/Entity Notation

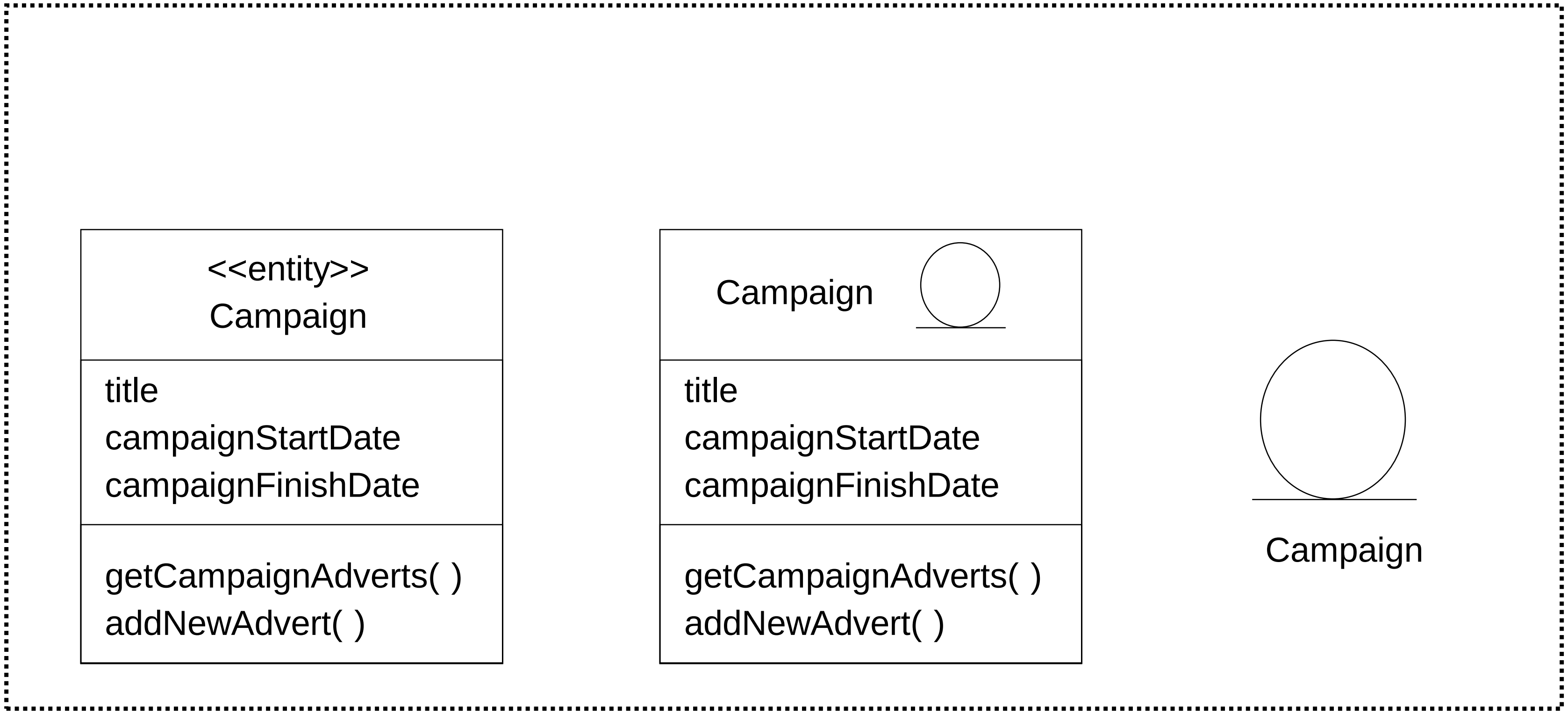
Different <<stereotypes>>/icons for BCE class types



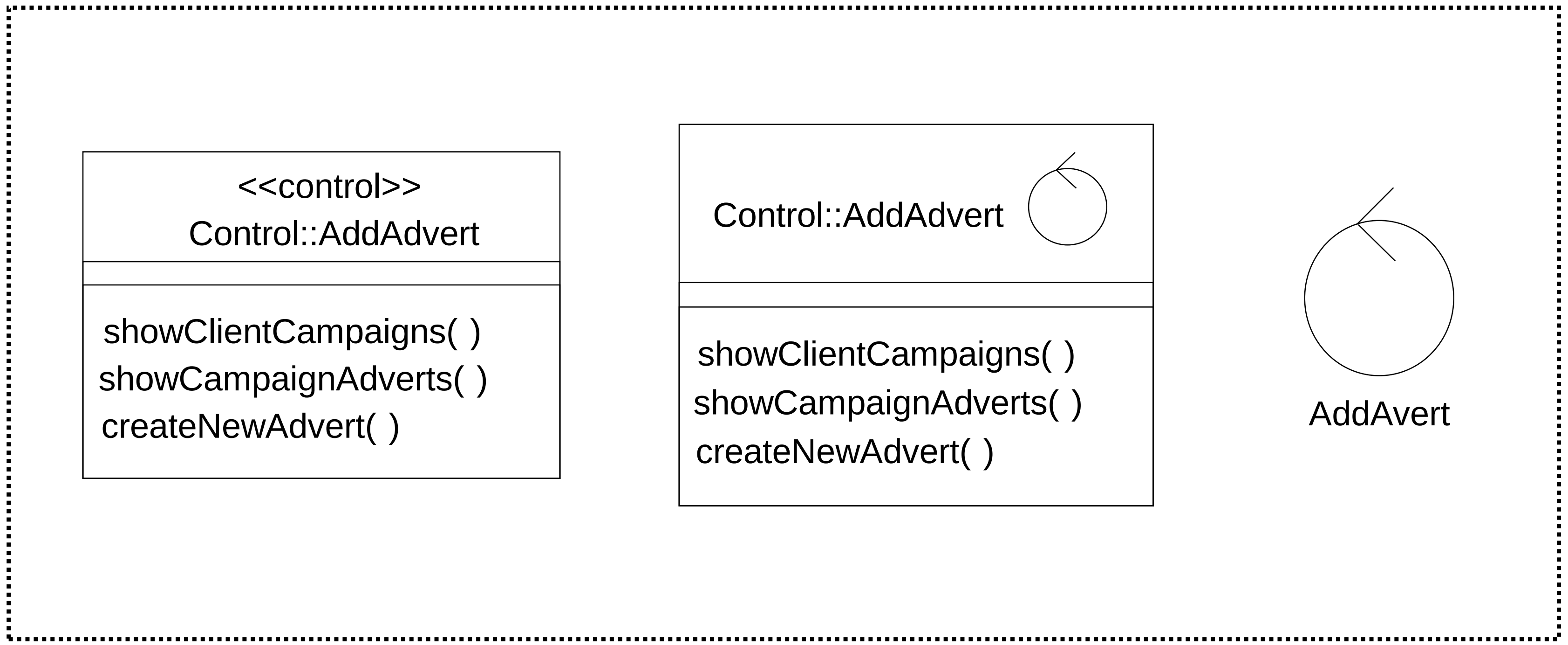
Class Diagram: Alternative notations for boundary class:



Class Diagram: Alternative notations for Entity class:



Class Diagram: Alternative notations for Control class:



Four rules of communication apply to the three categories of classes:

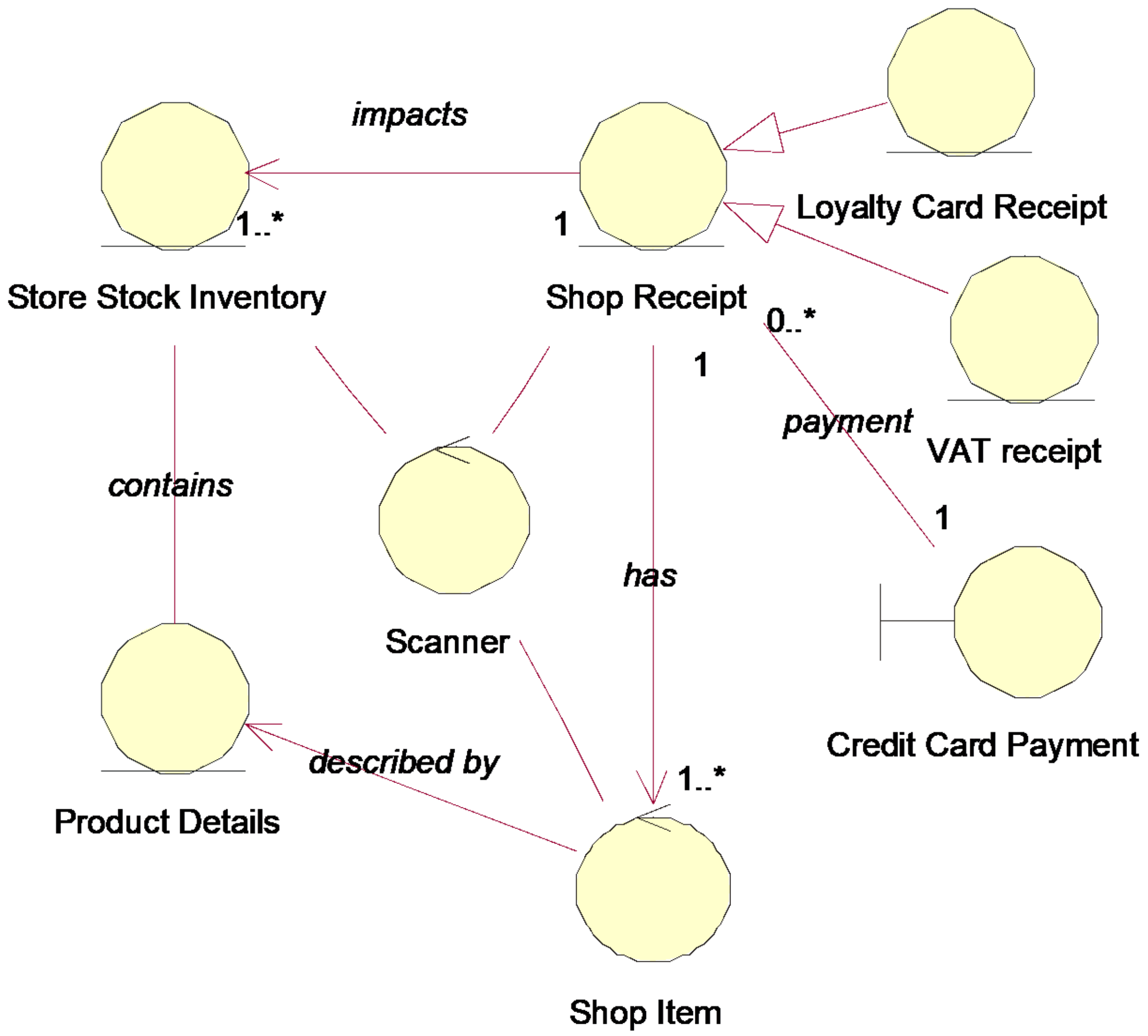
1. Actors can only talk to boundary objects.

2. Boundary objects can only talk to controllers and actors.

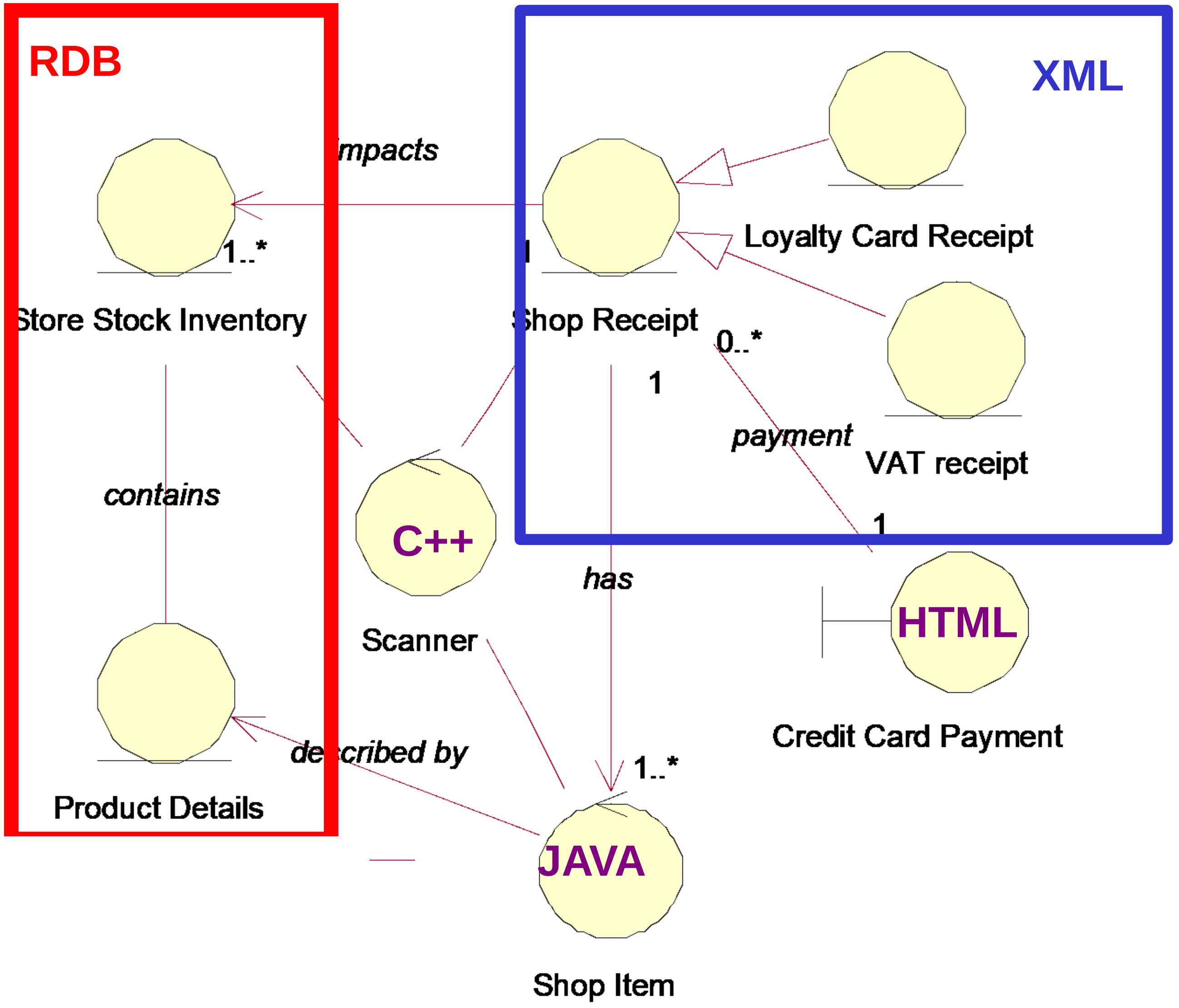
3. Entity objects can only talk to controllers.

4. Controllers can talk to boundary objects and entity objects, and to other controllers, but not to actors

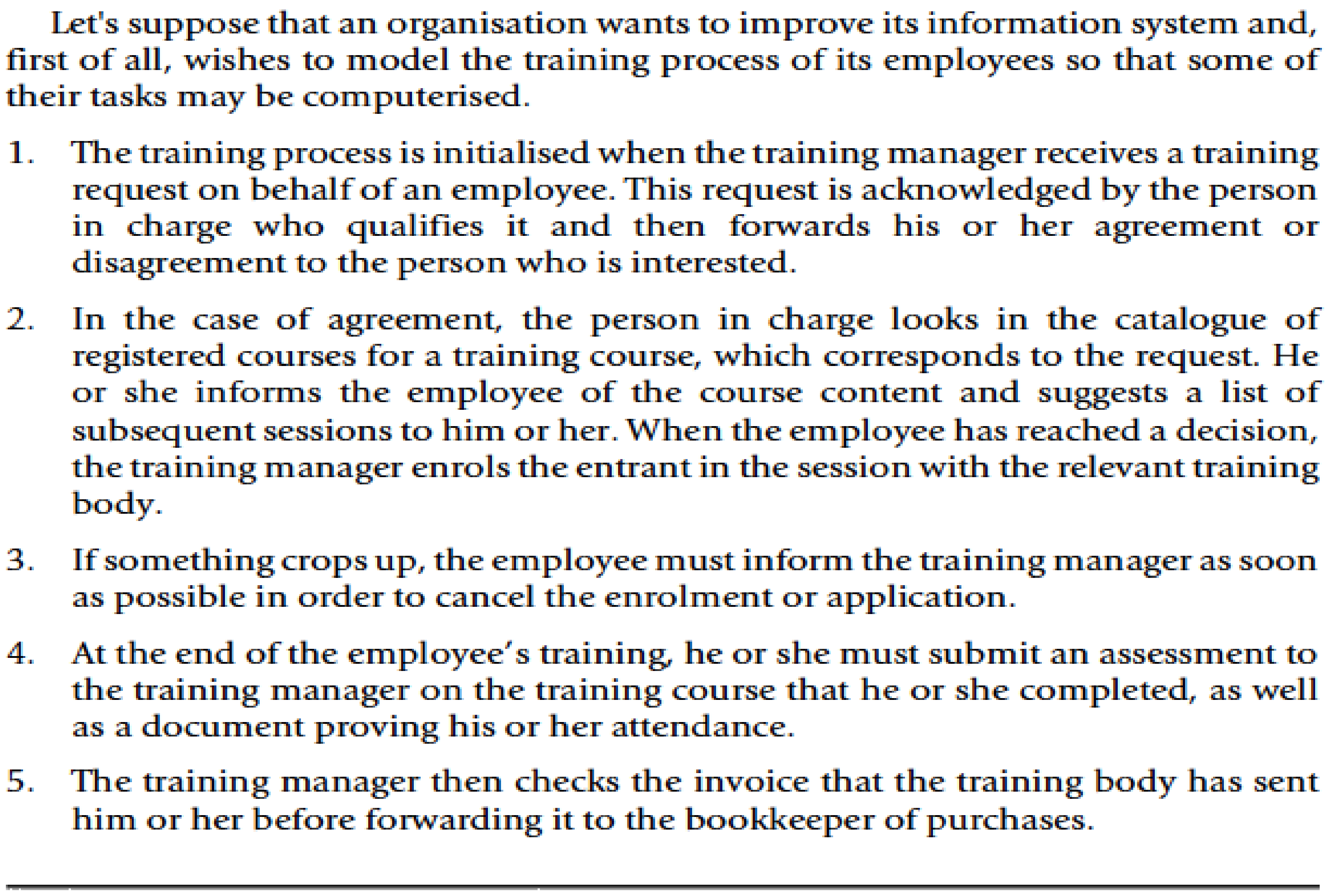
Class Diagram- BCE approach using ICON notation



UML is Technology Neutral



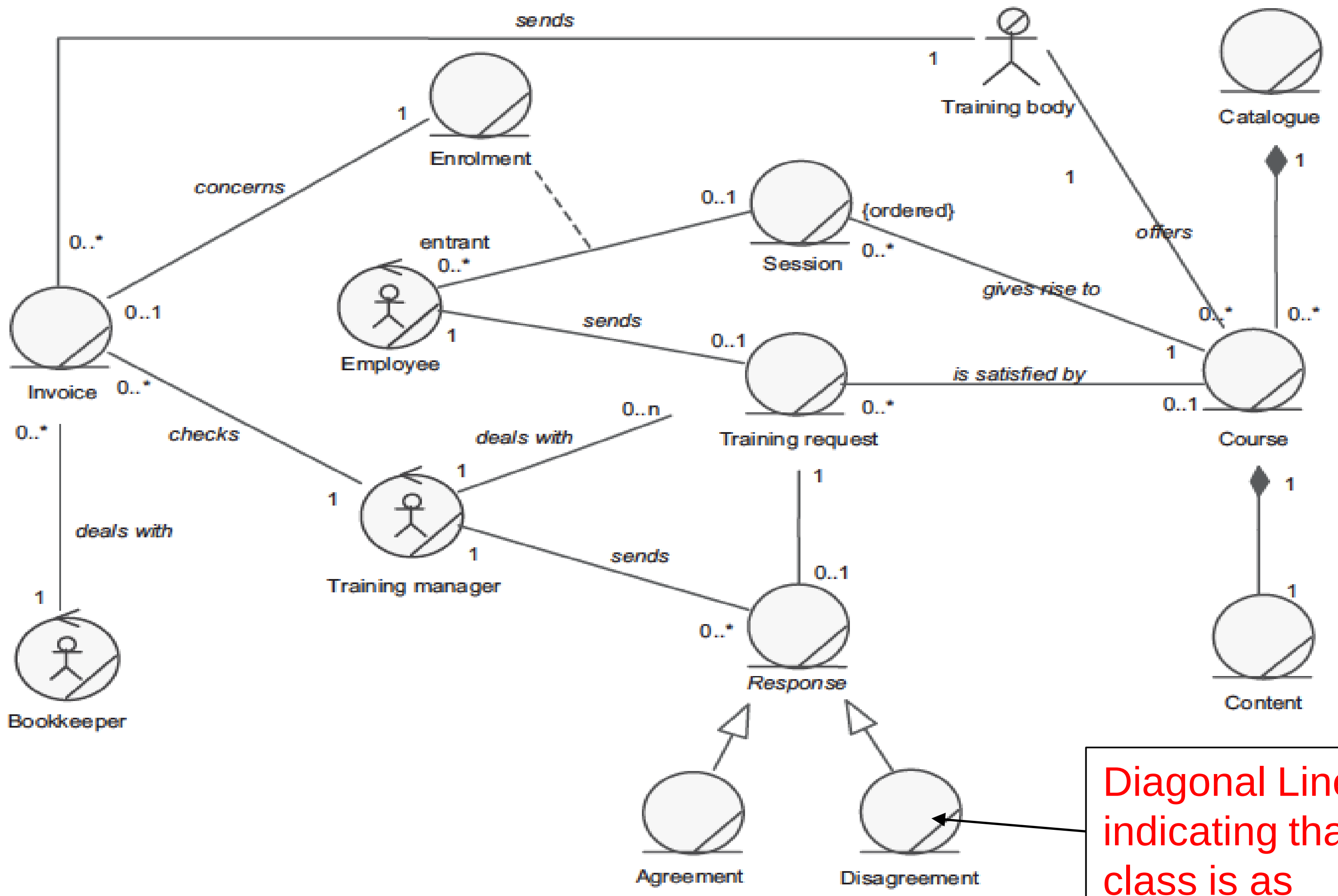
a) Draw UML Class diagram that describes the key actors and information classes of the process below. Identify classes as boundary, control or entity.



Let's suppose that an organisation wants to improve its information system and, first of all, wishes to model the training process of its employees so that some of their tasks may be computerised.

1. The training process is initialised when the training manager receives a training request on behalf of an employee. This request is acknowledged by the person in charge who qualifies it and then forwards his or her agreement or disagreement to the person who is interested.
2. In the case of agreement, the person in charge looks in the catalogue of registered courses for a training course, which corresponds to the request. He or she informs the employee of the course content and suggests a list of subsequent sessions to him or her. When the employee has reached a decision, the training manager enrols the entrant in the session with the relevant training body.
3. If something crops up, the employee must inform the training manager as soon as possible in order to cancel the enrolment or application.
4. At the end of the employee's training, he or she must submit an assessment to the training manager on the training course that he or she completed, as well as a document proving his or her attendance.
5. The training manager then checks the invoice that the training body has sent him or her before forwarding it to the bookkeeper of purchases.

Partial Possible Solution



Diagonal Line indicating that this class is as business class

UML Quick Overview

You can model about 80% of problems using 20% of UML… that is intention in this module

Use case diagrams

- Describe the functional behavior of the system as seen by the user
- Used during requirements elicitation

Class diagrams

- Describe the static structure of the system: Objects, attributes, associations

Sequence and Activity diagrams

- Describe the dynamic behavior between objects of the system

OCL – Object Constraint Language

- Declarative technology-neutral Language to help in precision of model

That’s All Folks Thank You for Listening