University of Dublin
Trinity College

Information Modeling

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

In life: why use a Model? – student pov

Method of describing a concept

Testing an idea

Use as a guide

Simulating errors

Why Use a Model?

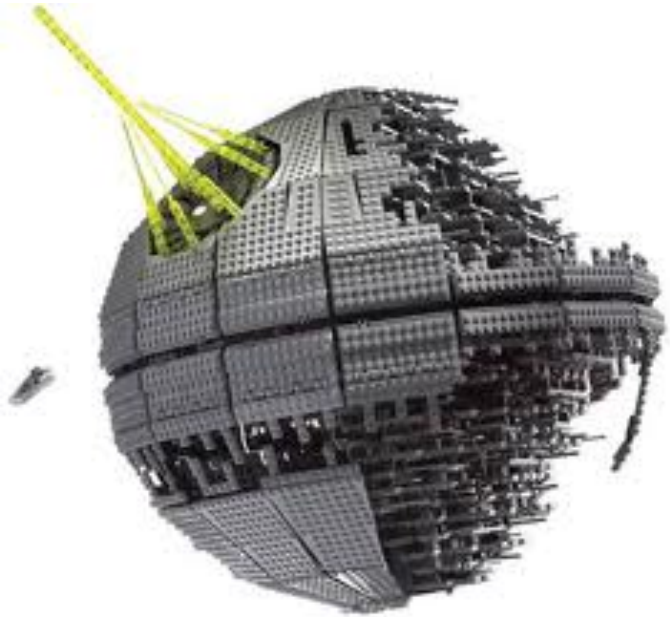
A model is quicker and easier to build

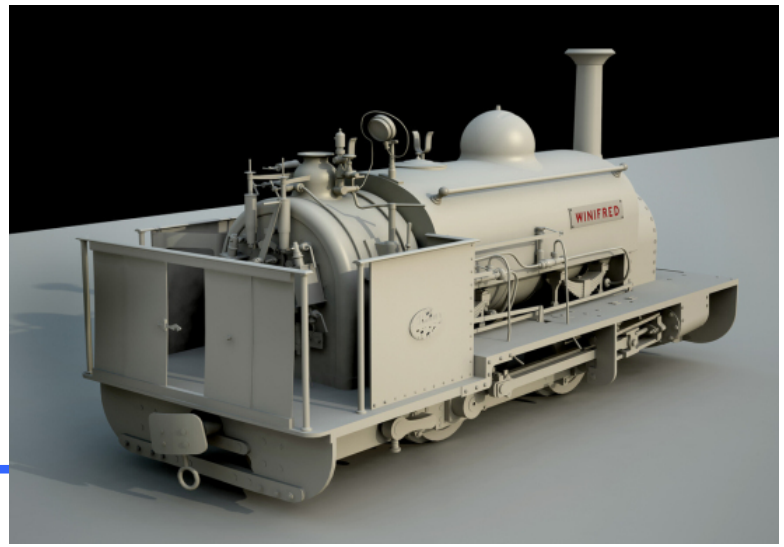
A model can be used in a simulation

A model can evolve as we learn

We can choose which details to include in a model

A model can represent real or imaginary things from any domain using any materials





Models in Traditional Engineering

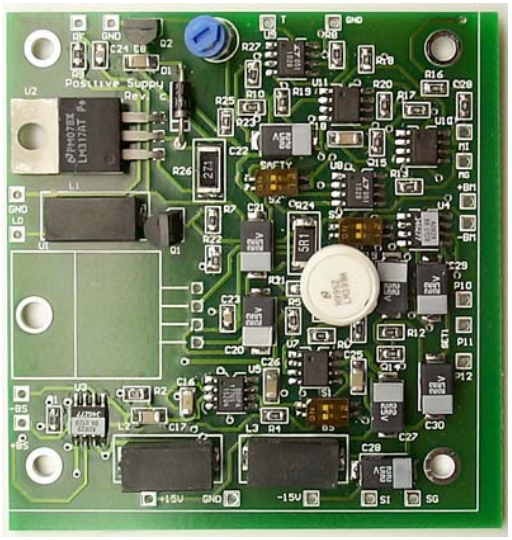
As old as engineering

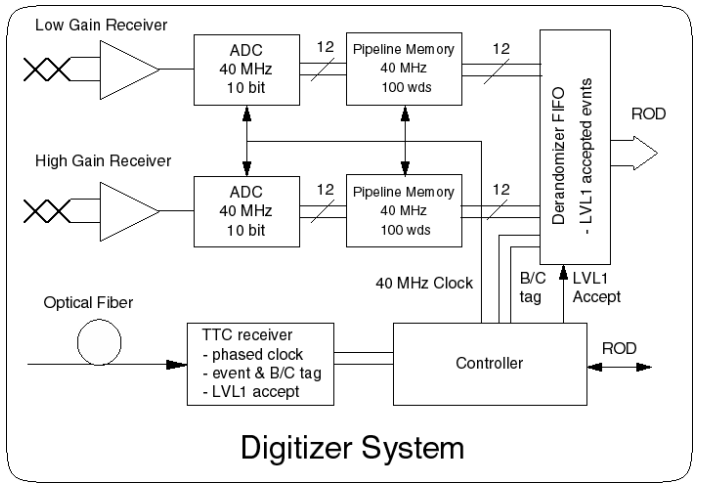
Traditional means of reducing engineering risk

Charles Parsons model steam turbine, 1880, Parsons Building TCD

Engineering Models

Engineering model: A reduced representation of some system that highlights the properties of interest from a given perspective



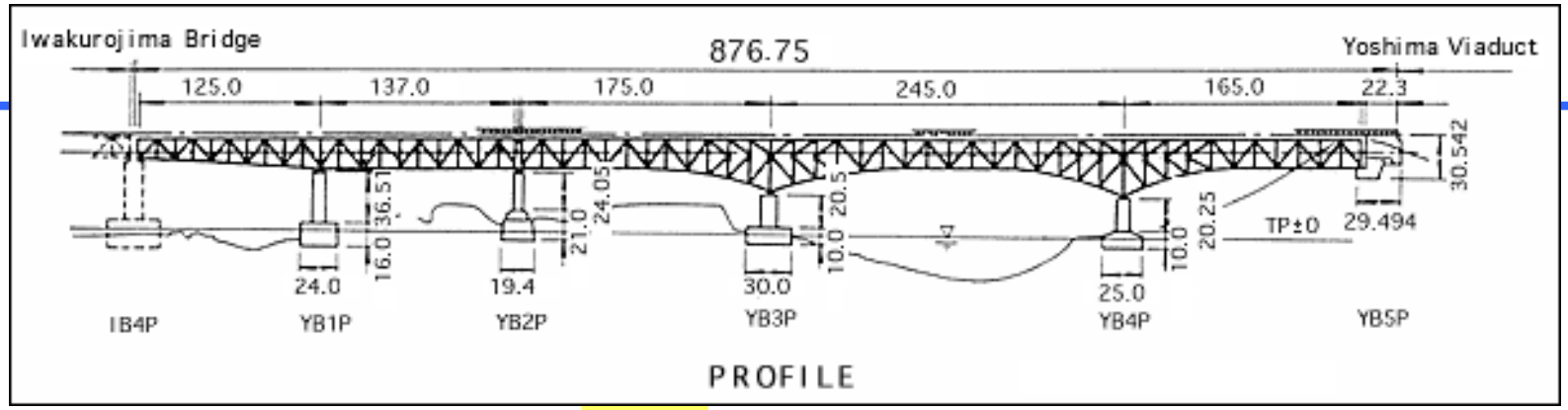


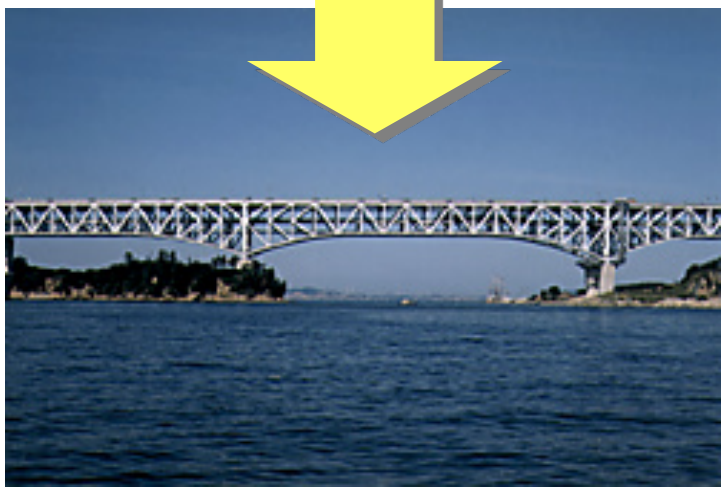
- We don’t see everything at once
- We use a representation (notation) that is easily understood

How Engineering Models are Used

1. To help us *understand* complex systems
   - Useful for both *requirements* and *designs*
   - Minimize risk by detecting errors and omissions early in the design cycle (at low cost)
     - *Through analysis and experimentation*
     - *Investigate and compare alternative solutions*
   - To communicate understanding
     - *Stakeholders: Clients, users, implementers, testers, documenters, etc.*
2. To *drive implementation*
   - The model as a blueprint for construction

A Common Problem with Engineering Models





***Semantic Gap* due to:**

- **Idiosyncrasies of actual construction materials**
- **Construction methods**
- **Scaling effects**
- **Skill sets**
- **Misunderstandings**

Can lead to serious errors and discrepancies in the realization

Small Exercise

Close Laptops

Using pen and paper **only**, figure out a way to communicate a summary of the info on the next slide simply and efficiently to someone else

The Learnalot University

The LearnAlot University offers a number of degrees to under graduate and post graduate students who may be fulltime or parttime.. The educational structure of the university consists of schools. Schools contain several departments. While a single school administers each degree, the degree may include courses from other schools. In fact the university prides itself on the freedom of choice given to students in selecting courses towards their degrees.
Each university degree has a number of compulsory courses and a number of elective courses. Each course is at a given level and has a credit point value. ……

…. A student's proposed program of study is entered in the online enrolment system. The system checks the program's consistency, checks if courses are open and reports any problems……

Small Exercise

Now Swap page with another person and look at their summary

Who succeeded in communicating all information?

Real World: Non trivial problem
Example Relational Schema

Information Model

An information model in software engineering is a representation of concepts, relationships, constraints, rules, and operations to specify data semantics for a chosen domain of discourse.

It can provide sharable, stable, and organized structure of information requirements for the domain context

Typically Technology Neutral

Most approaches to modelling have a methodology for mapping to concrete information/data implementations (e.g. relational database, XML etc.)

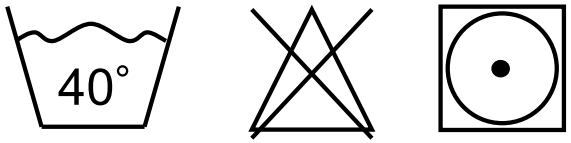
Why use Diagrams in modelling?

In software engineering, like in other engineering disciplines, we use diagrams to represent models …

Abstract shapes are used to represent things or actions from the real world, or from a system in our case …

Diagrams follow rules or standards

The standards make sure that different people will interpret the diagram in the same way



Terminology

A **system** is the overall thing that is being modelled

A **sub-system** is a part of a system consisting of related elements

A **model** provides complete set of views on the information for use by different stakeholders

A model may consist of a single diagram, but most consist of a set of diagrams and supporting data and documentation

What Different Stakeholders and Views? – student pov

Stakeholders	Views
Client	structure
Developer	Dynamic
Investors	Goals
Maintainer	Characteristics of info
Management	Consequences
Researchers	
Data protectiion	

Different Stakeholders and Views

- Client stakeholder
- Information Architect
- Database Designer
- Software Engineer
- Deployer
- End User
- Data Analyst
- Application Programmer
- Quality Manager
- Maintainer

Typical views are

- Structural/Static view– how data and information is associated, what tasks does the information support etc.
- Dynamic behaviour view
- Physical deployment view
- Model management view – ongoing management of models during lifecycle

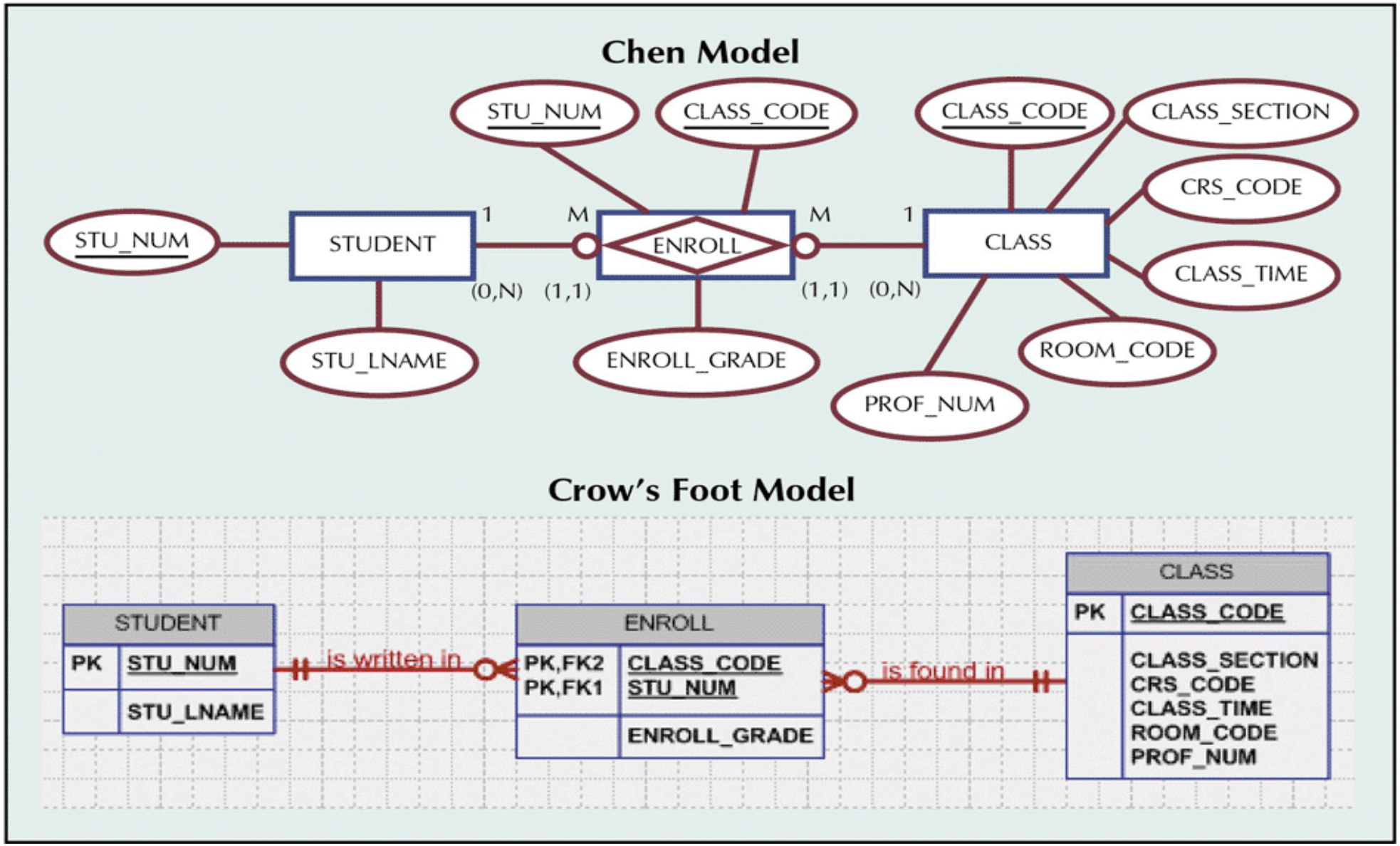
Example approach that supports info model structure view: **Entity Relationship Diagrams**

ER Diagrams widely accepted and adopted graphical approach to conceptual modeling for database design

However, there are several markedly different variations of ERD notations

An ERD depicts (some of) the ER model's entity types, attributes and relationships, and (depending on the diagram style) varying amounts of other info such as connectivities, cardinalities, keys, weakness, …

Quick Flavour of Two Styles of Diagram



Example approach that supports info model dynamic view: **Data Flow Diagrams**

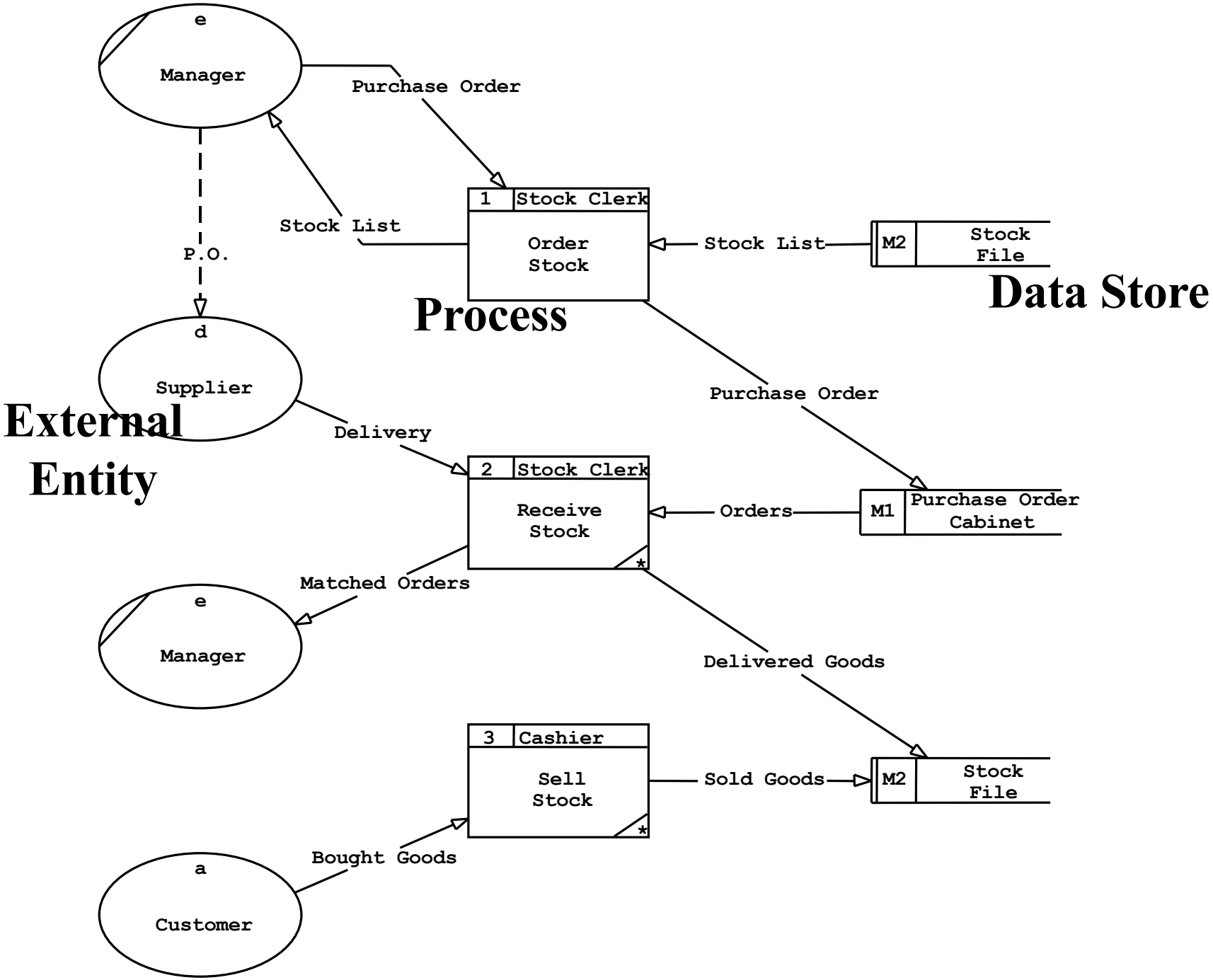
Data Flow Modelling is a widely used and mature analysis technique, and is recommended by most structured methods

However there are variations of notations in use

Data Flow Models (DFMs) are easy to understand and, with a little practice, reasonably quick and straightforward to develop

They consist of two parts: a set of Data Flow Diagrams (DFDs) and a set of associated textual descriptions

Provides an effective method to support understanding processing of information in a system


e
Manager
Purchase Order
Stock List
P.O.
1
Stock Clerk
Order
Stock
Stock List
M2
Stock
File
Data Store
Process
d
Supplier
External
Entity
Delivery
Purchase Order
2
Stock Clerk
Receive
Stock
Orders
M1
Purchase Order
Cabinet
*
Matched Orders
e
Manager
Delivered Goods
3
Cashier
Sell
Stock
Sold Goods
M2
Stock
File
*
Bought Goods
a
Customer

What is UML and what is it not?

Specification and design language, not a programming language! Modeling language.

- Mostly visual but has precise semantics
- Diagrams consist of well-defined elements (graphical) and have rules on how to use and combine elements
- Abstract syntax, well-formedness rules, semantics can be found in the official documentation

UML is not a software tool

UML is not a methodology but only a notation

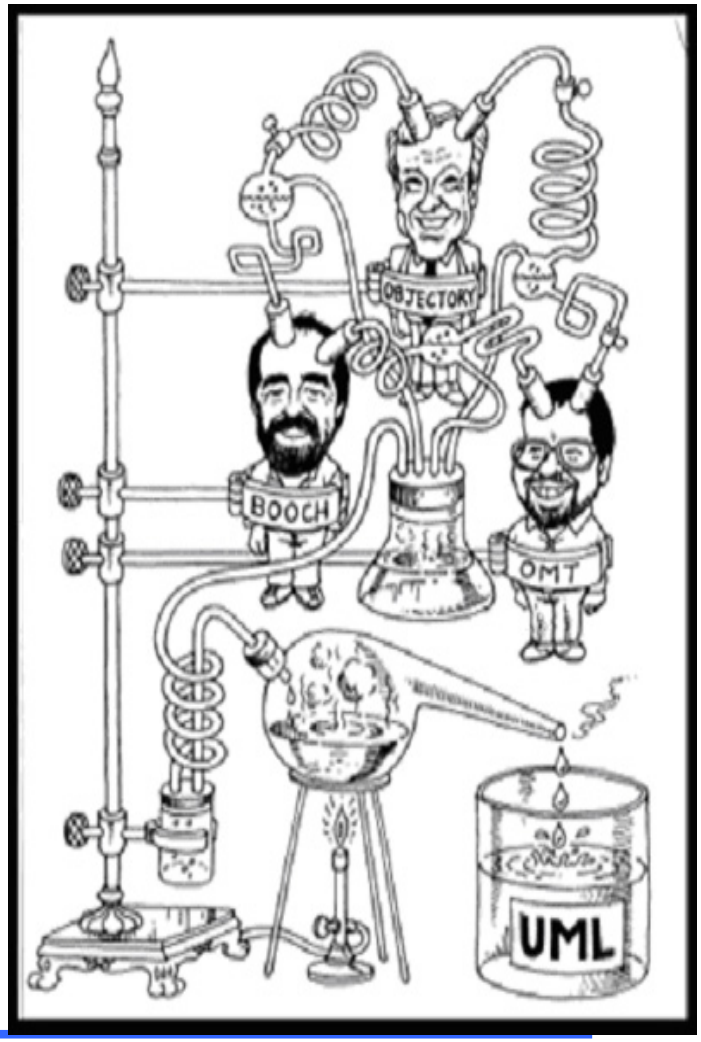
Origins of UML

Object-oriented Analysis and Design techniques

UML's goal is to bring together the best features of all notations to produce an industry standard

Managed by the Open Management Group

- http://www.omg.org/gettingstarted/what_is_uml.htm
- current version 2.5 (March 2015)
  - http://www.omg.org/spec/UML/2.5/PDF



Why use UML for Information Modeling

De facto standard for object-oriented modeling, maintained by a standards organisation OMG

Bring good ideas in consistent framework, supported by many tools, profiles, methods and processes exist

Diagrams help visualize designs and cope with complexity

Implements or support the principles of:

- Abstraction
- Separation of concerns
- Modularity
- Rigor and formality

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

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

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

UML views and diagrams cont.

Major View	Sub-view	Diagram	Concepts
physical	deployment	deployment diagram	artifact, dependency, manifestation, node
model management	model management	package diagram	import, model, package
	profile	package diagram	constraint, profile, stereotype, tagged value

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

Summary so far

Models reduce risk.

Unified Modeling Language – Blueprints for designing software systems. (a bit bloated though)

Diagrams are used to express design in a common format.

Models are used to analyze the design.

Structure diagrams show static architecture.

Dynamic diagrams show how components interact.

University of Dublin
Trinity College

UML Use Cases

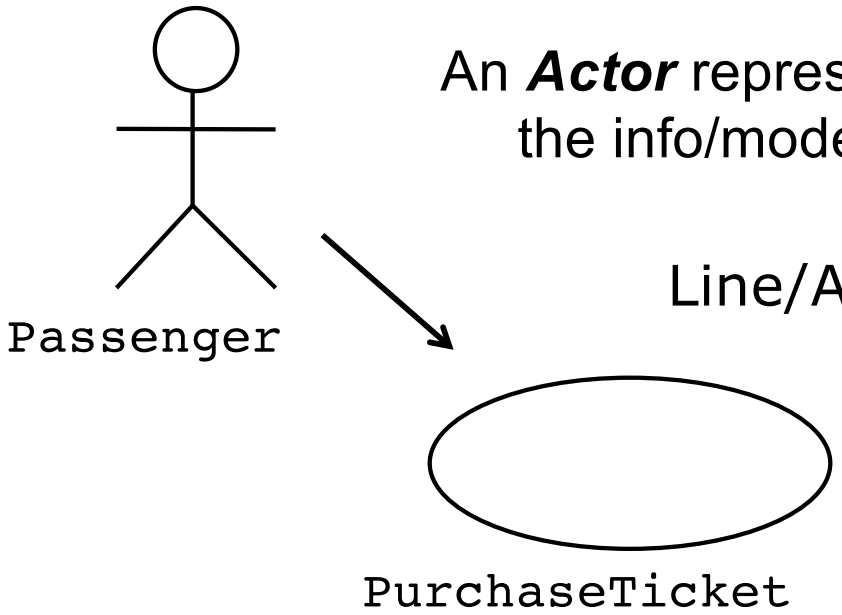
What are UML **Use Case Diagrams** used for?

With the help of a use case diagram, you can communicate:

- The scope of the **tasks your information model supports** and **for whom** (people, organizations, or external systems)

UML Use Case Diagrams

Used during requirements elicitation and analysis to represent external behavior ("visible from the outside of the system")



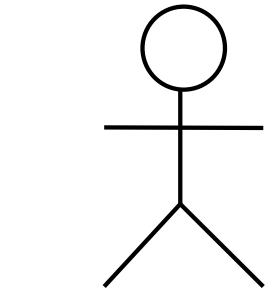
An ***Actor*** represents a role, that is, a type of user of the info/model system

Line/Arrow, who involved in interaction

A single *use case* represents a class of functionality provided by the system

Use case model:
The set of all use cases that completely describe the functionality of the system.

Actors



An actor is a model for an external entity which interacts (communicates) with the info model/system:

- User
- External system (Another system)
- Physical environment (e.g. Weather)

An actor has a unique name and an optional description

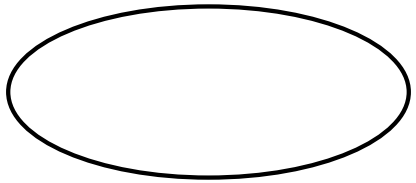
Examples:

- **Passenger**: A person for the train
- **Credit Card System**: An external system that provides the system with financial API

Description

Name

Use Case



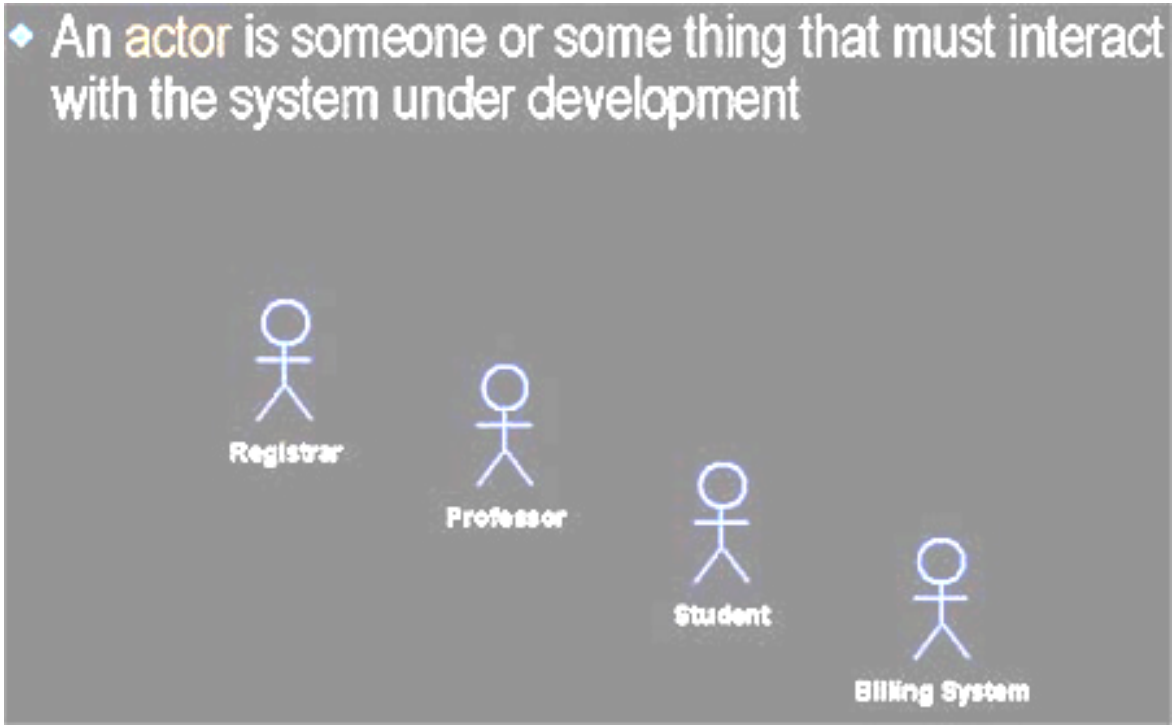
`PurchaseTicket`

- A use case represents a class of functionality provided by the system
- Use cases are also described textually, with a focus on the event flow between actor and system
- The textual use case description consists of 6 parts:
  1. Unique name
  2. Participating actors
  3. Entry conditions
  4. Exit conditions
  5. Flow of events
  6. Special requirements.
- Note that a rectangle is used to show what use cases are in the scope/boundary of system/info model

Modelling what the system will do: Use Case Diagram Step 1

Identify each actor

- Drawn as “stick person”
- For example in for a University Info System



Modelling what the system will do: Use Case Diagram Step 2

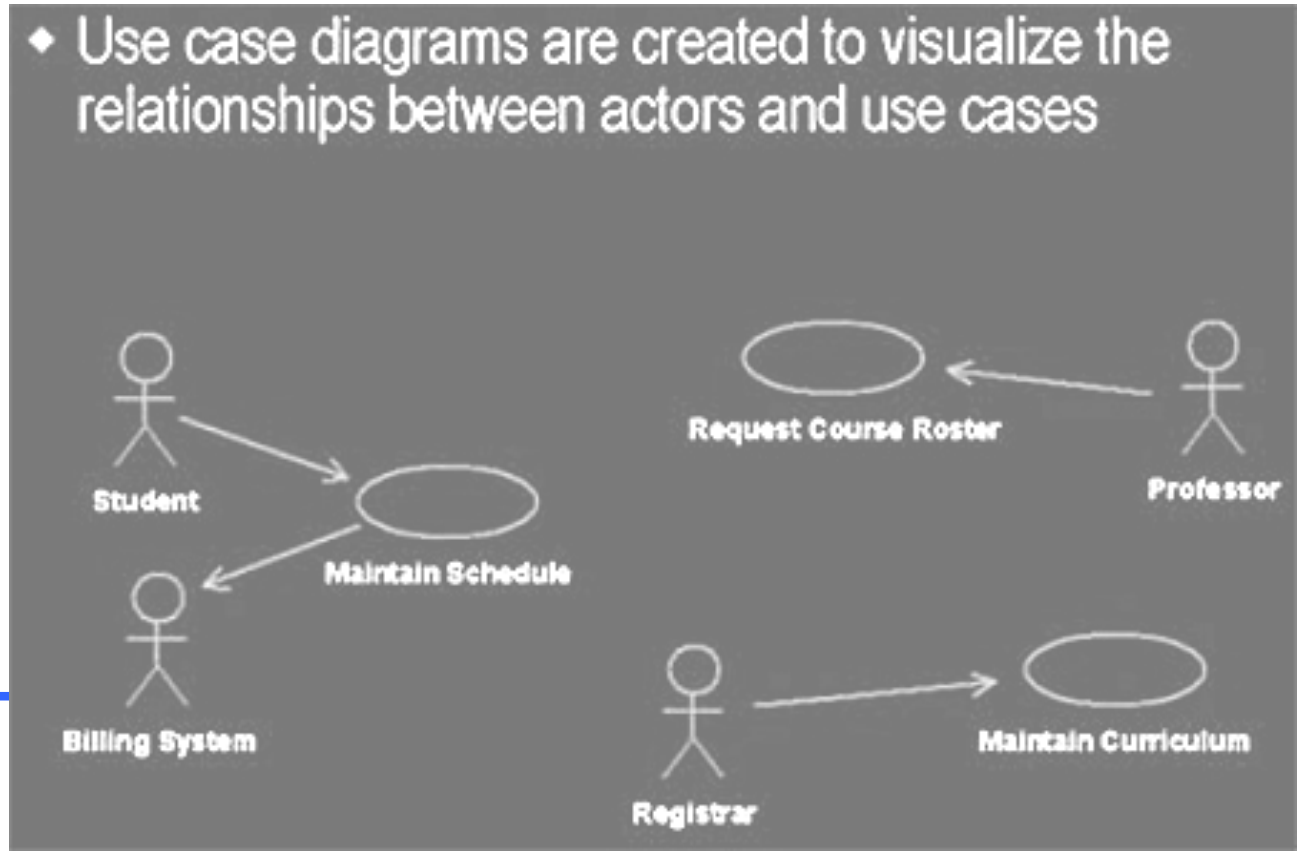
Identify each Use Case

- Drawn as Oval
- A transaction performed by the system for the actor in a dialog
  - something that provides value to the actor.
- How find them?
  - Ask yourself why would the actor want to use the system
  - Examples
    - *Professor wants to have list of students*
    - *Student wants to enrol in a module*
    - *Billing system wants to issue appropriate fees to student*
- Be careful naming them: Use ACTION verbs

Modelling what the system will do: Use Case Diagram Step 3

Draw use case diagram

- Link actors to use cases through relationship arrows
- **Direction of arrow** indicates who can initiate the interaction, no arrow indicates either can initiate



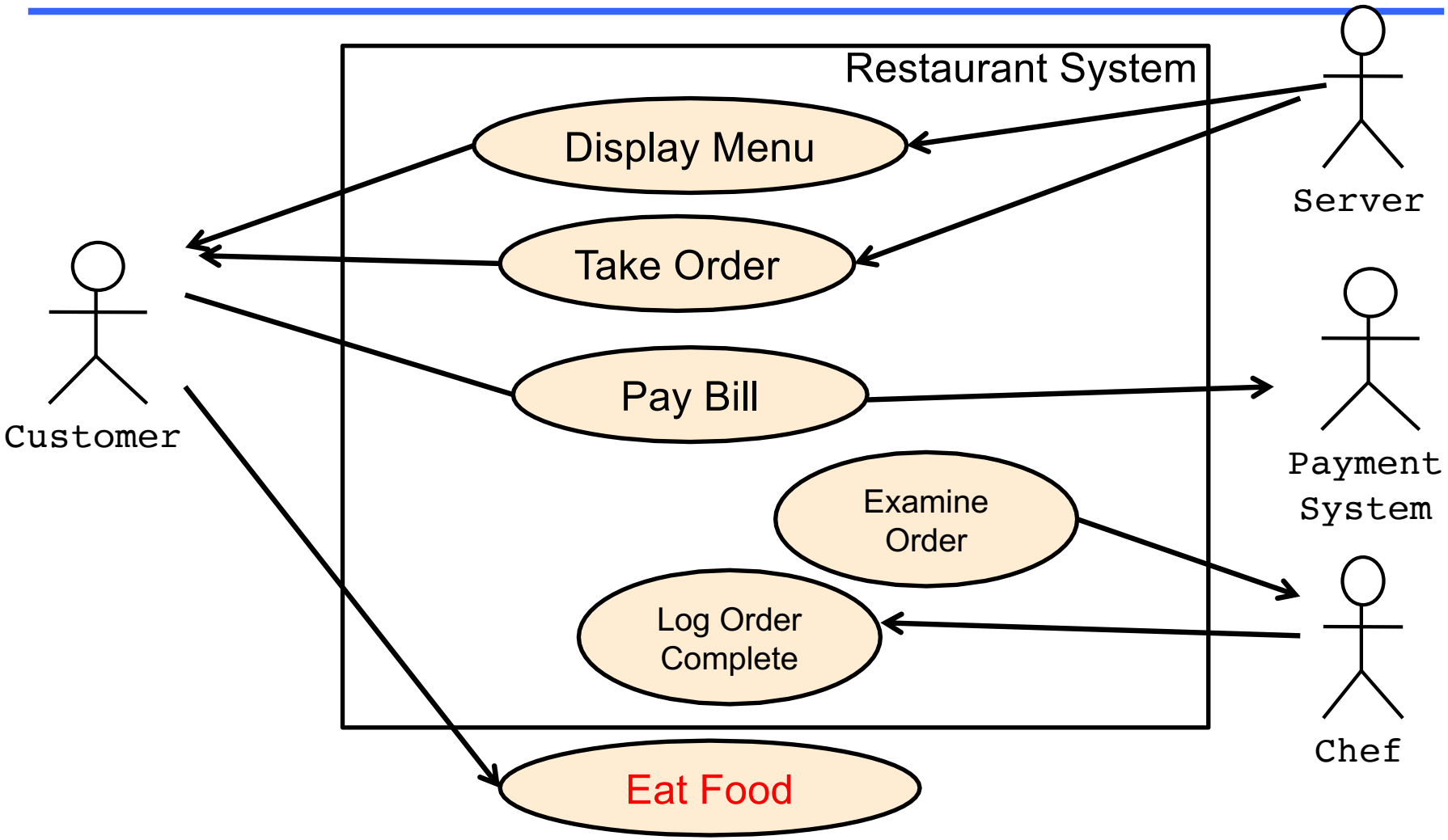
YOUR TURN TO PUT INTO ACTION!

Imagine that you need to create an Information Model to support a waiter with an iPad app in a **Restaurant**

1. Identify a couple of the actors
2. Identify a couple of the use cases

(Remember each use case represents a functionality/service that the system provides/supports for the actor)

One Possible Solution



Note that we can model "Eat Food" if we like in our diagram BUT as the INFO SYSTEM does not support it, it is drawn outside the box

Modelling what the system will do: Use Case Diagram Step 4

Consider relationships between use cases

Extends Relationship <<extends>>

- To represent seldom invoked use cases or exceptional functionality

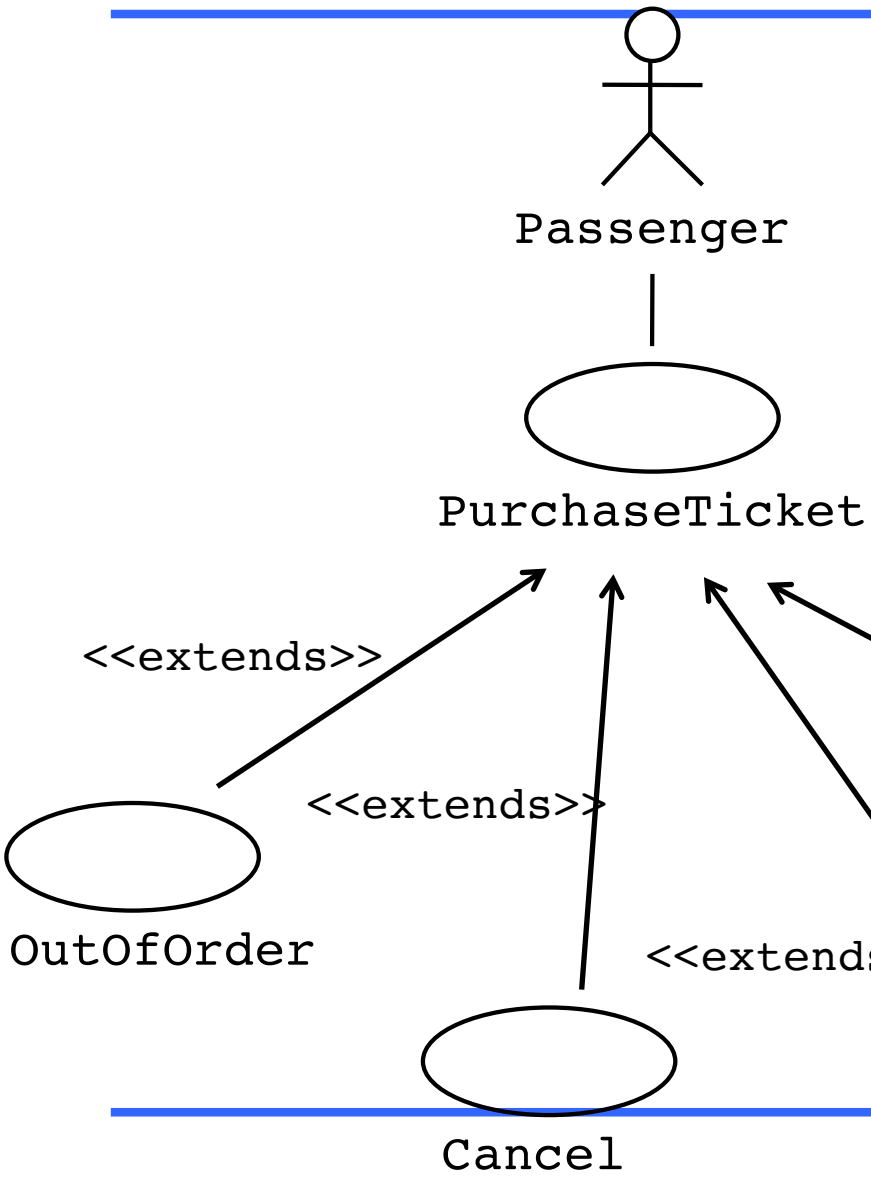
Includes Relationship <<includes>>

- To represent functional behavior common to more than one use case.

Use Case Generalisation (triangular arrow)

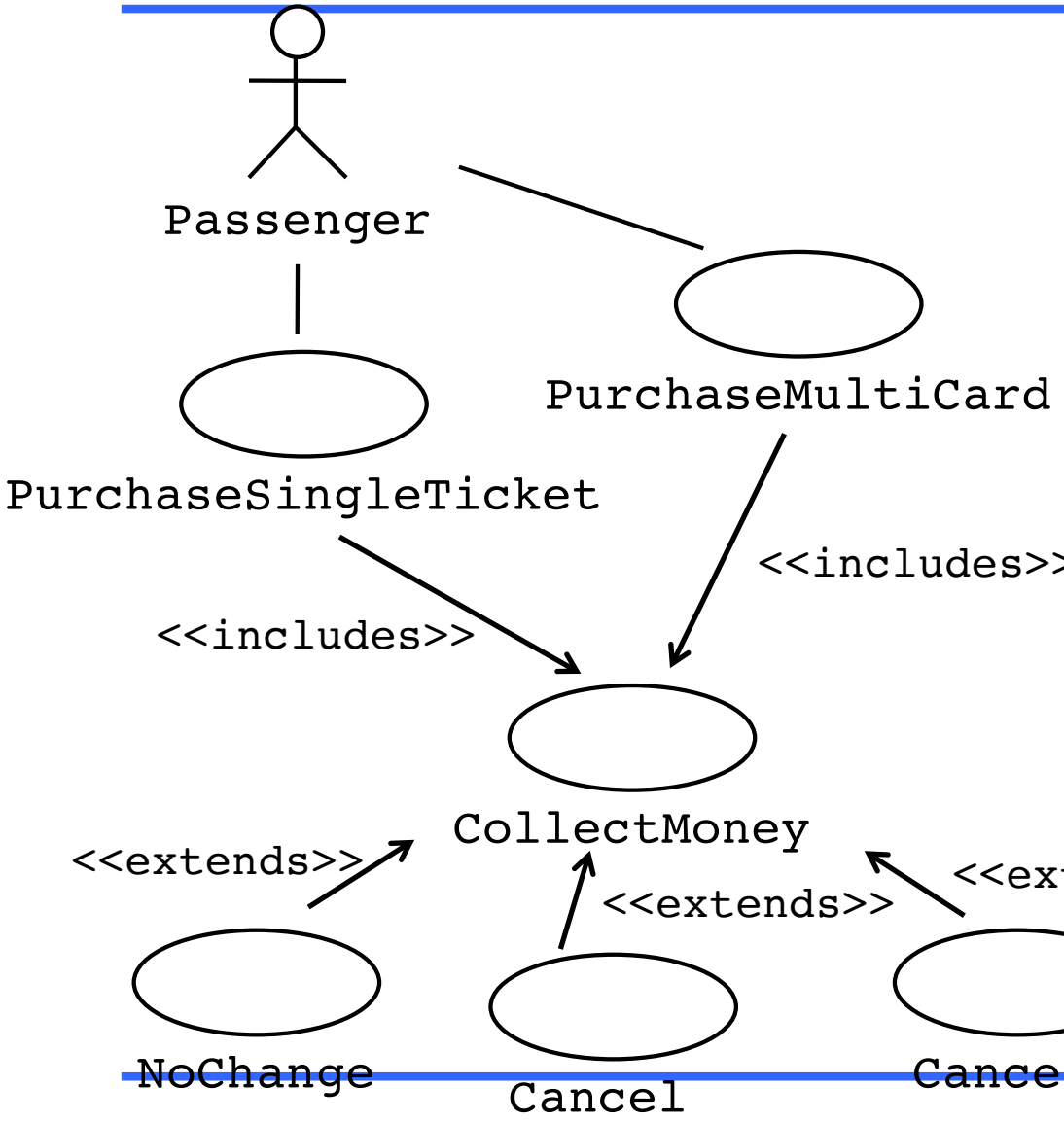
- A relationship between a general use case and a more specific use case that inherits and adds features to it

The <<extends>> Relationship



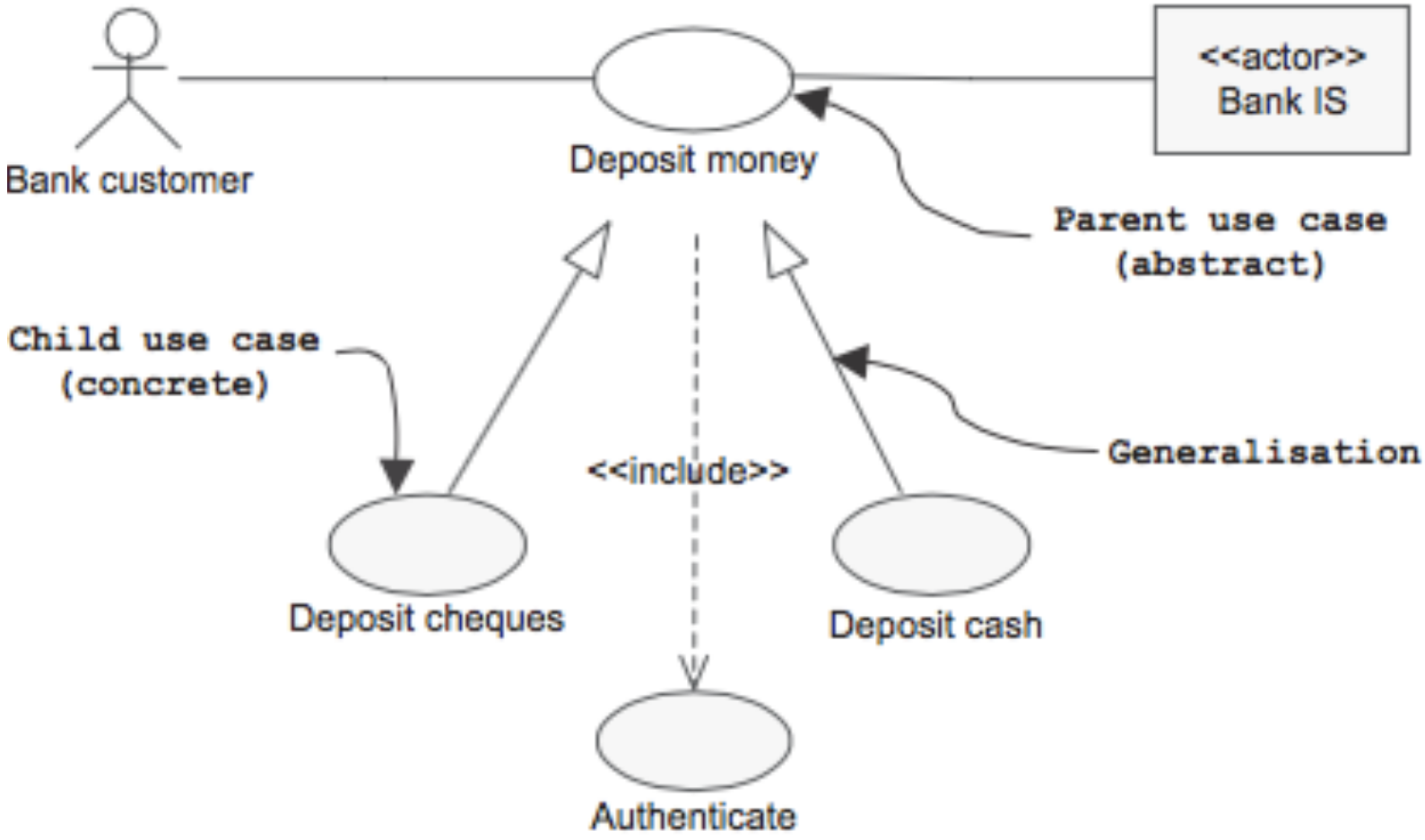
- <<extends>> relationships model exceptional or seldom invoked cases
- The exceptional event flows are factored out of the main event flow for clarity
- The direction of an <<extends>> relationship is to the extended use case
- Use cases representing exceptional flows can extend more than one use case.

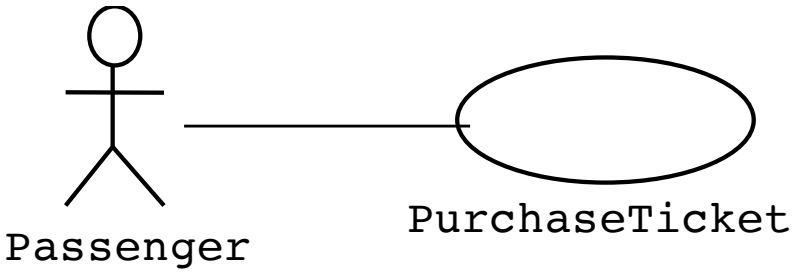
The <<includes>> Relationship



- <<includes>> relationship represents common functionality needed in more than one use case
- <<includes>> behavior is factored out for reuse, not because it is an exception
- The direction of a <<includes>> relationship is to the using use case (unlike the direction of the <<extends>> relationship).

Use Case Generalisation Example





Modelling what the system will do: Use Case Descriptions Step 6

1. Name: `Purchase ticket`

2. Participating actor:
`Passenger`

3. Entry condition:

`Passenger` stands in front of ticket distributor

`Passenger` has sufficient money to purchase ticket

4. Exit condition:

`Passenger` has ticket

5. Normal Scenario:

1. `Passenger` selects the number of zones to be traveled
2. Ticket Distributor displays the amount due
3. `Passenger` inserts money, at least the amount due
4. Ticket Distributor returns change
5. Ticket Distributor issues ticket

6. Error Scenario:

Some money inserted of wrong type

- Return money and provide explanation.

YOUR TURN TO PUT INTO ACTION! – WORK IN PAIRS

Modelling what the system will do: Use Case Diagram Step 4

Consider relationships between use cases

Extends Relationship <<extends>>

- To represent seldom invoked use cases or exceptional functionality

Includes Relationship <<includes>>

- To represent functional behavior common to more than one use case.

Use Case Generalisation (triangular arrow)

- A relationship between a general use case and a more specific use case that inherits and adds features to it

CS2041: Use Case Exercise

Name:
Student Number:

- From the problem statement below
  1. Identify Actors, Use Cases and draw use case diagram
  2. Write a textual description for “Process Sale” use case, (a) for a normal scenario and (b) for an error scenario

PROBLEM STATEMENT

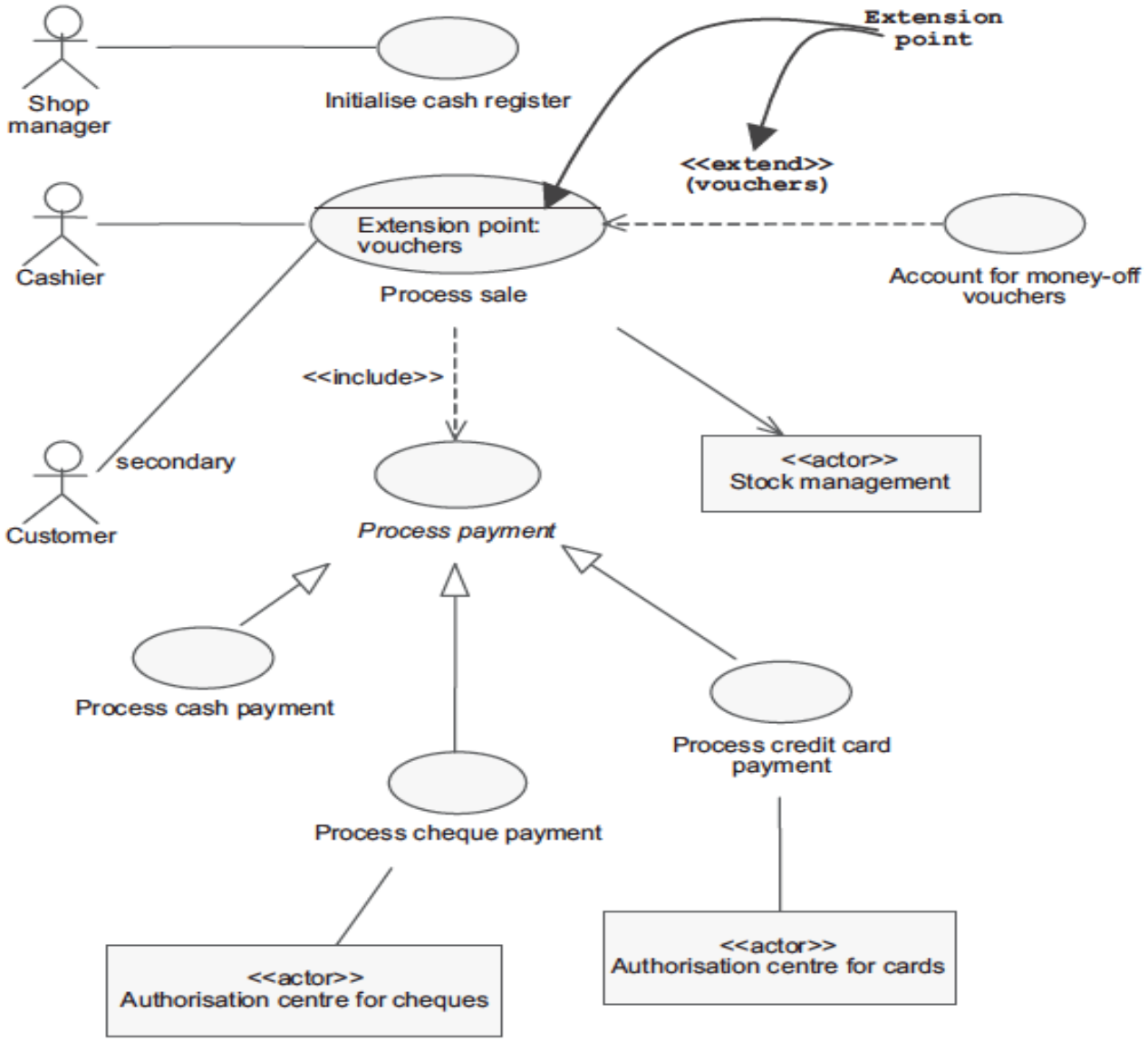
The standard procedure of using a cash register is as follows:

- A customer arrives at the checkout to pay for various items
- The cashier records the bar code number of each item, as well as the quantity if it is greater than one.
- The cash register displays the price of each item and its description.
- When all the purchases are recorded, the cashier indicates the end of the sale.
- The cash register displays the total cost of the purchases.
- The customer selects his or her payment method:
  - Cash: the cashier takes the money from the customer and puts it in the cash register, the cash register indicates how much change the customer is to be given;
  - Cheque: the cashier verifies that the customer is financially solvent by sending a request to an authorisation centre via the cash register;
  - Credit card: a banking terminal forms part of the cash register. It sends a request for authorisation to an authorisation centre, according to the card type.
- The cash register records the sale and prints a receipt.
- The cashier gives the receipt to the customer.

Once the items have been entered, the customer can present money-off vouchers for certain items to the cashier. When the payment transaction is finished, the cash register sends the information on the number of items sold to the stock management system.

Every morning, the shop manager initialises the cash registers for the day.

Partial Cashier Possible Solution



Title: Process sale **Type**: detailed essential

Summary: a customer arrives at the checkout with the items he or she would like to purchase. The cashier records the items and collects payment. At the end of the transaction, the customer leaves with the items.

Actors: Cashier (primary), *Customer (secondary)*.

Creation date: 05/17/02 **Date of update**: 11/10/02

Version: 1.1 **Person in charge**: Pascal Roques

Normal Scenario: Process Sale

Preconditions:

- The cash register is open; a checkout assistant is signed on to it.

Main success scenario:

1. This use case starts when a customer arrives at the checkout with items that he or she would like to purchase.
2. The cashier records each item. If there is more than one of the same item, the cashier also indicates the quantity.
3. The cash register establishes the price of the item and adds the information on the item to the sale in progress. The cash register displays the description and the price of the item in question.
4. Once the cashier has recorded all the items, he or she indicates that the sale is finished.
5. The cash register calculates and displays the total amount of the sale.
6. The cashier informs the customer of the total amount.
7. The customer chooses a payment method:
   a. In the case of cash payment, execute the "Process cash payment" use case;
   b. In the case of credit card payment, execute the "Process credit card payment" use case;
   c. In the case of cheque payment, execute the "Process cheque payment" use case.
8. The cash register records the sale that has been carried out and prints a receipt.
9. The cashier gives the cash register receipt to the customer.
10. The customer leaves with the items he or she has purchased.

Error Scenario: Process Sale

E1: customer is unable to pay

The E1 sequence starts at point 1 of the main success scenario.

2. The customer does not have enough cash to pay for the items.
3. The cashier cancels the whole sale and the use case fails, or the customer pays using another payment method (Cf. "Process cheque payment", or "Process credit card payment").

E2: cashier is unable to give change

The E1 sequence starts at point 4 of the main success scenario.

5. The cash register drawer does not contain enough change in order to give the customer the money he or she is owed.
6. The cashier asks his or her supervisor for more change, or suggests to the customer that he or she pay using a different payment method (Cf. "Process cheque payment", or "Process credit card payment").

Reminder

Do some Research on the organisation

Decide on aspect of operations information modelling is going to support

- Email me a paragraph including rationale for that focus and the kind of things you things you think you will need to be modelling
- ONE email per group to me by 7pm Wednesday 4th October. In subject header you MUST include Group Number in the Subject line of Email

Develop UML Use Cases of Information Model

Prepare for presentation Thursday 12th October

- problem being tackled
- research done
- UML Use Cases diagrams
- example of scenario text for 1 Use Case
- strengths & weaknesses of design

Deadline for presentation material

- Email by Wednesday 11th October 5pm
- PDF version to be of presented
- PDF version that includes presentation with speaker notes
- You MUST include Group Number in the Subject line of Email

Thursday 5th October

Prof. Dave Lewis

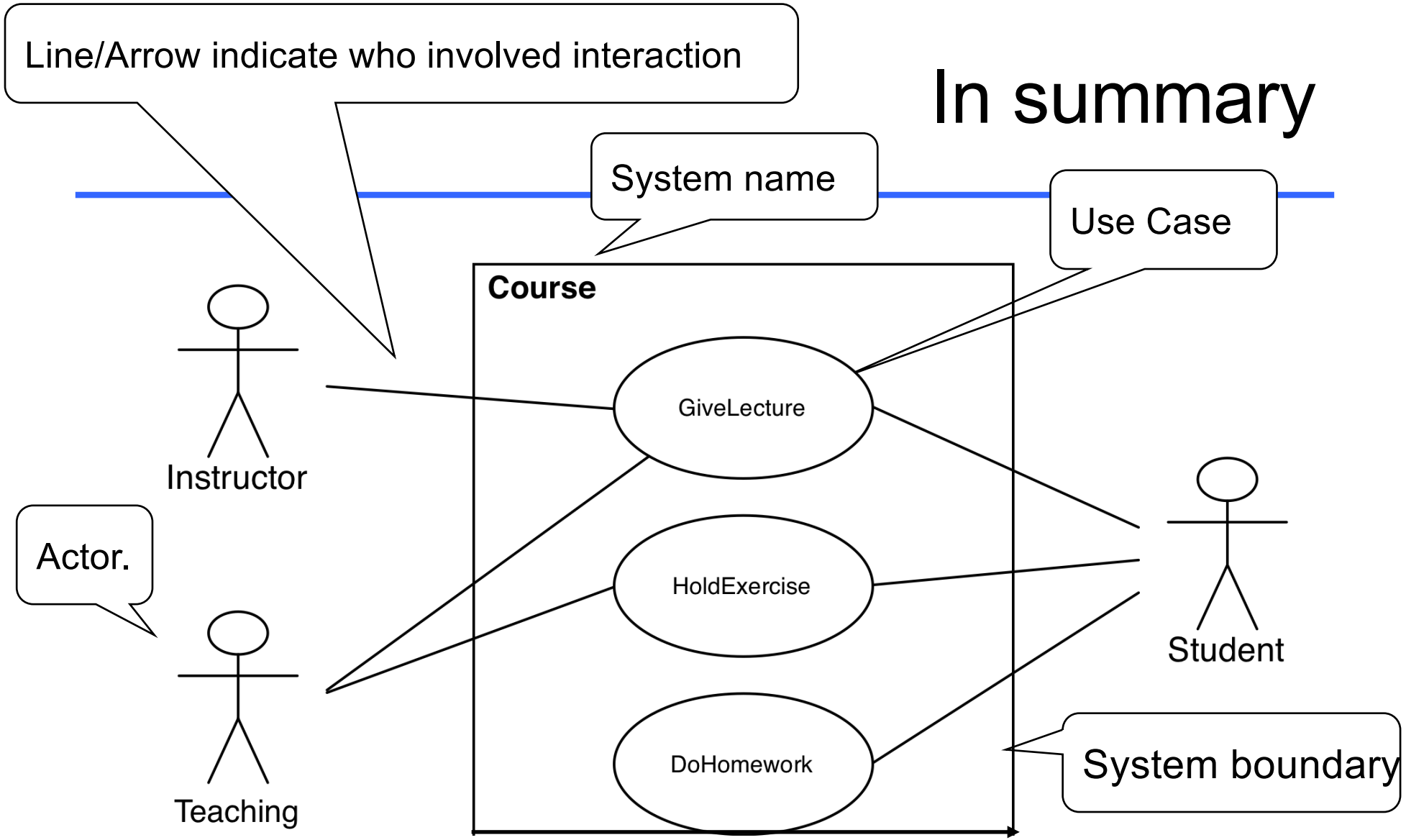
Motivation behind Ethics Canvas

Using the Ethics Canvas in a development project

For use within Group Project

Check out https://ethicscanvas.org

In summary



Use case diagrams represent the functionality of the system from user's point of view