# Review of diagrams

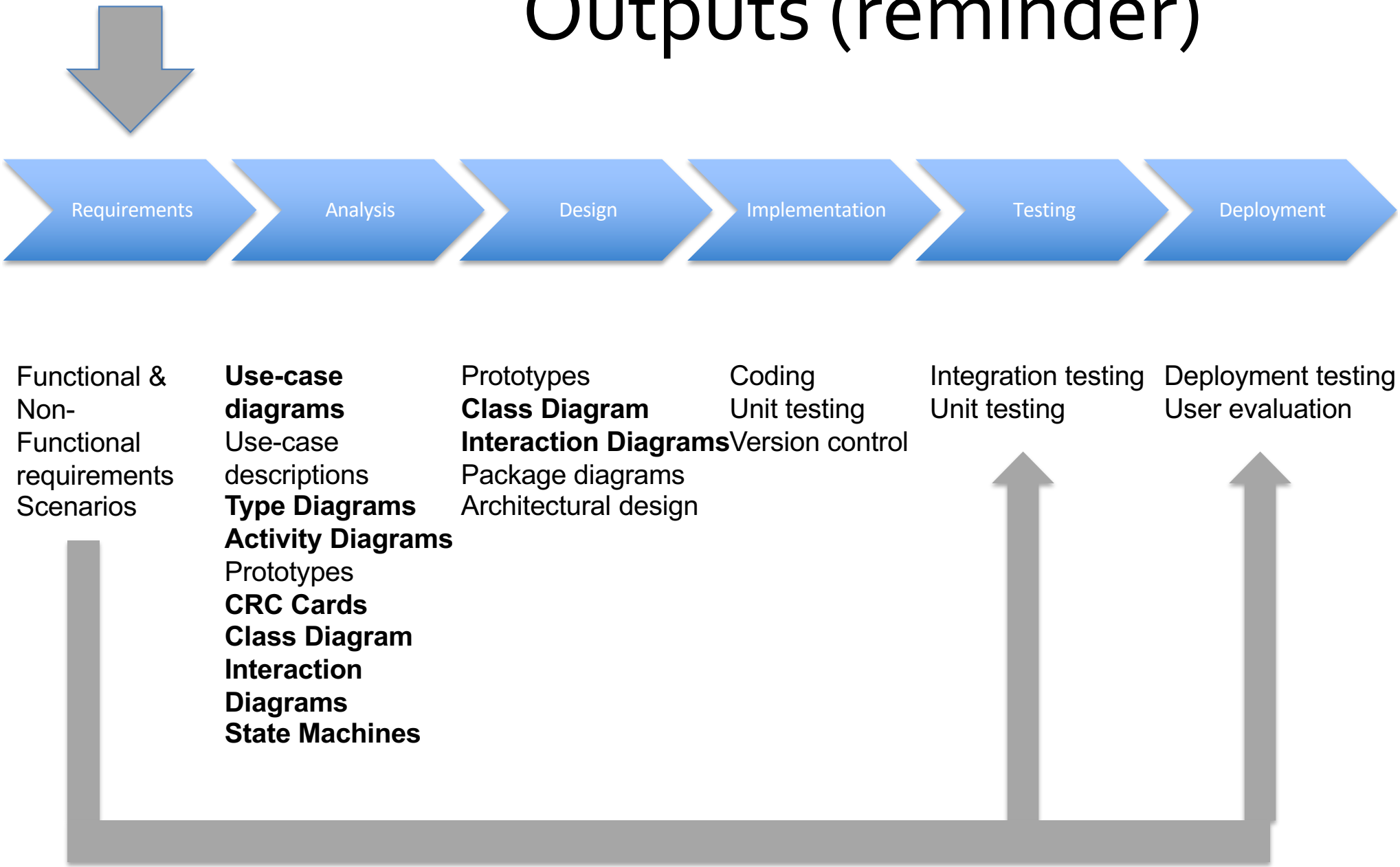
Week 7 class

CE202 Software Engineering

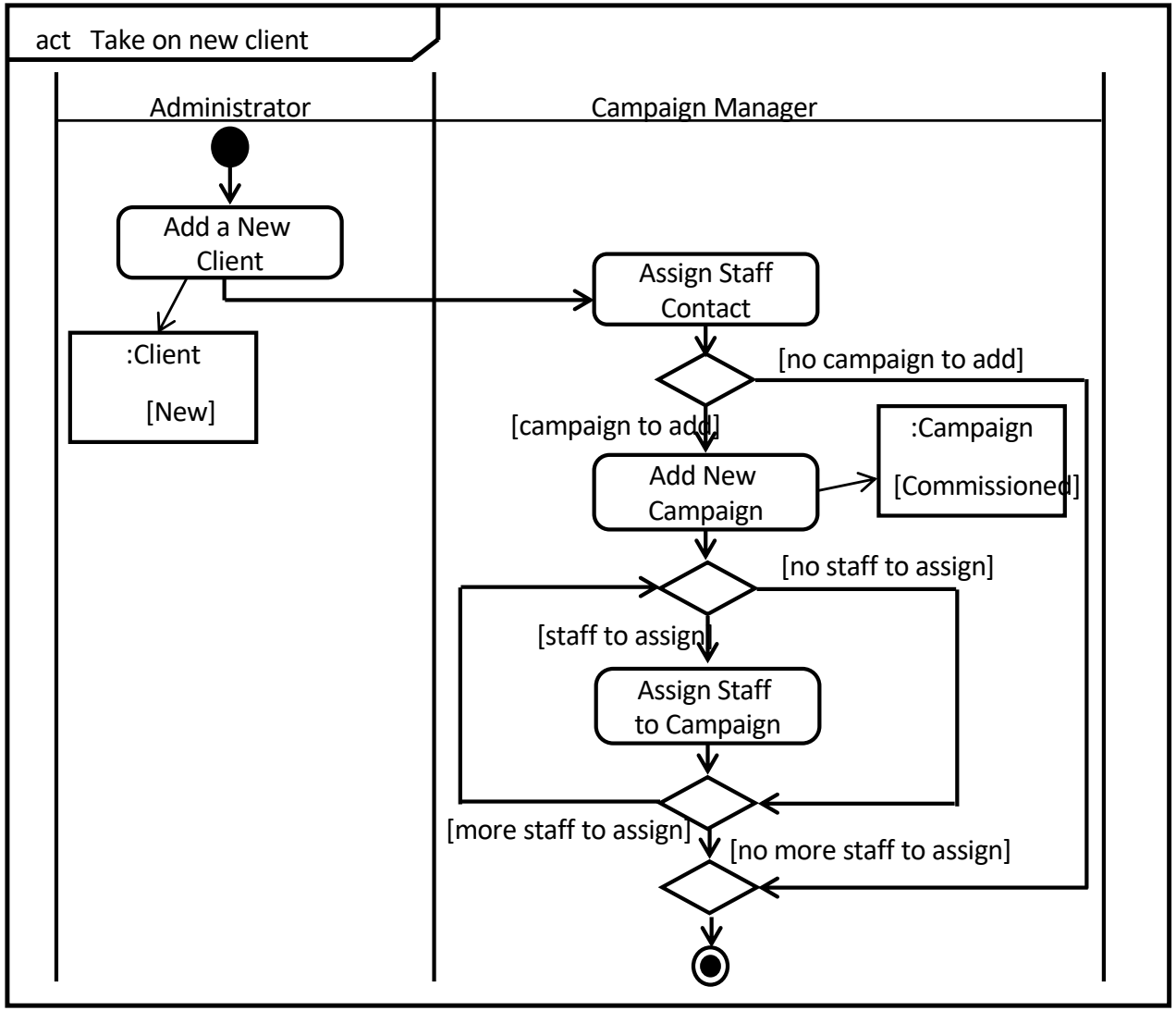
M. Gardner

# Waterfall Lifecycle and Main Outputs (reminder)

Problem Definition
Viewpoints/Scope



# Activity diagrams




**Can be used at any stage in the lifecycle. Examples:**

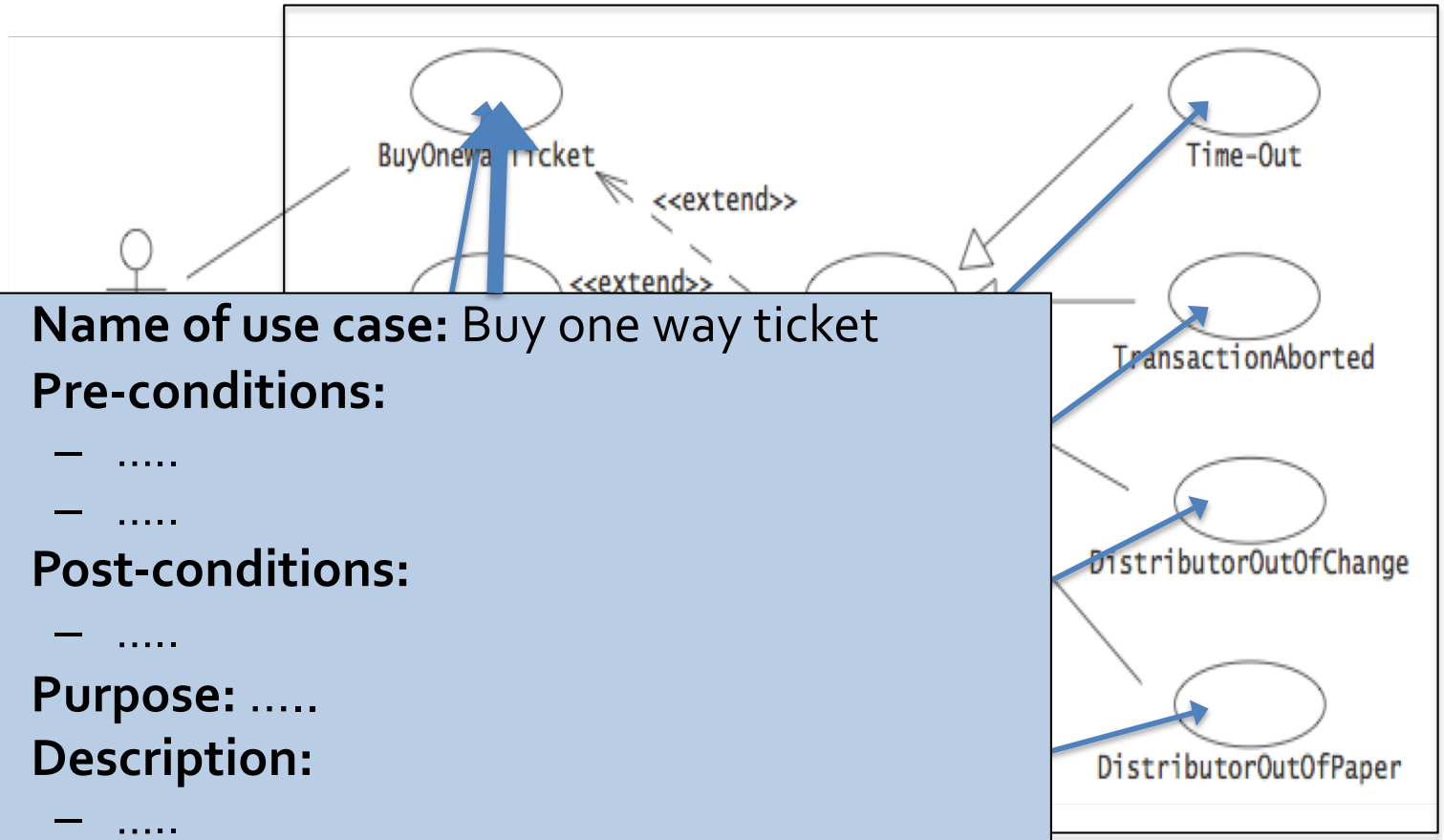
- Model a process derived from the requirements
- Model logic flow for a use case
- Model logic flow for code
- Etc

**Need to ensure consistency with other diagrams. Eg:**

- Use of objects
- Use-cases (if activities are based on them)

The decision about whether to use this diagram is dependent on your particular problem. Eg. whether you need to model a process/activity flow?

# Use case diagram




- **Name of use case:** Buy one way ticket
- **Pre-conditions:**
  - .....
  - .....
- **Post-conditions:**
  - .....
- **Purpose:** .....
- **Description:**
  - .....
  - .....
  - .....
  - .....
  - Etc .....

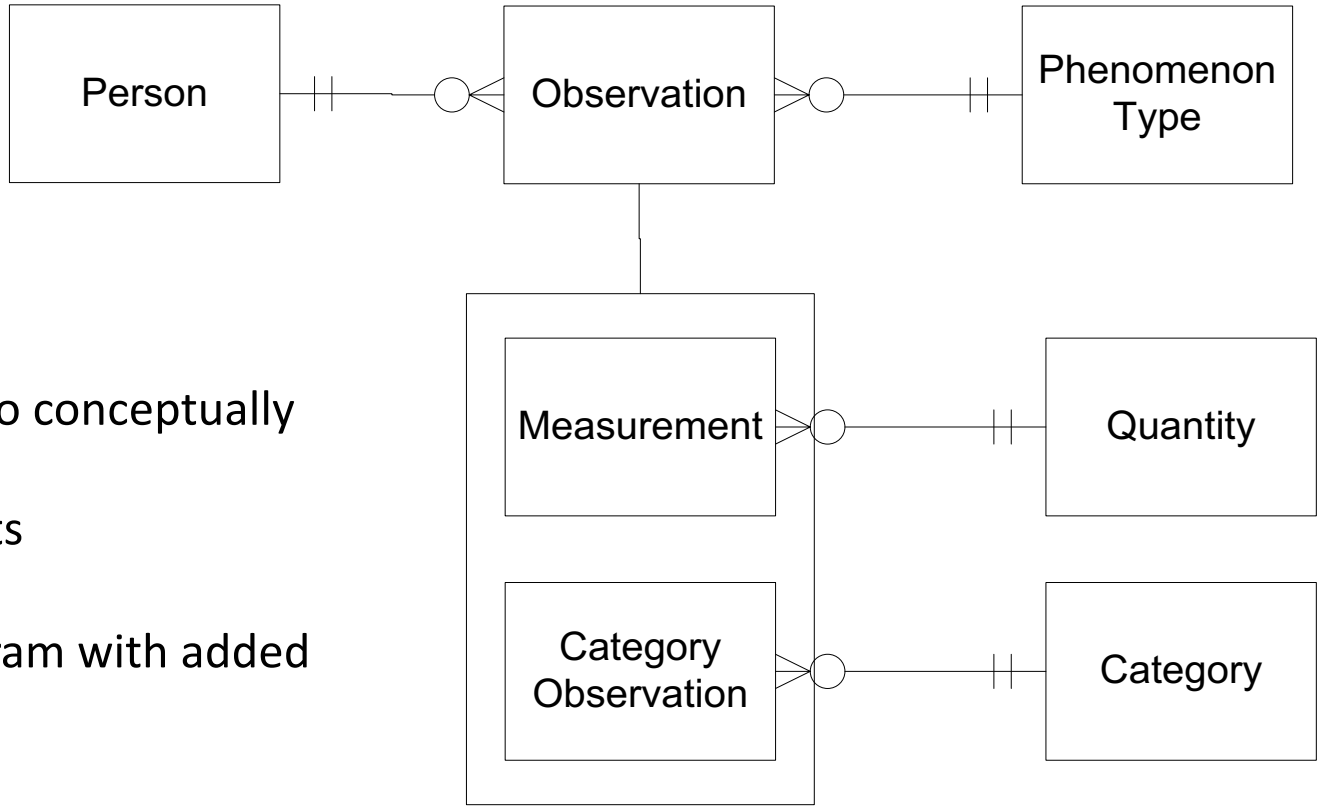
Each **use case bubble** should link to a detailed **use case description**

Relationships between USE CASES

- <<extend>>
- <<include>>
- <<generalise>> or △

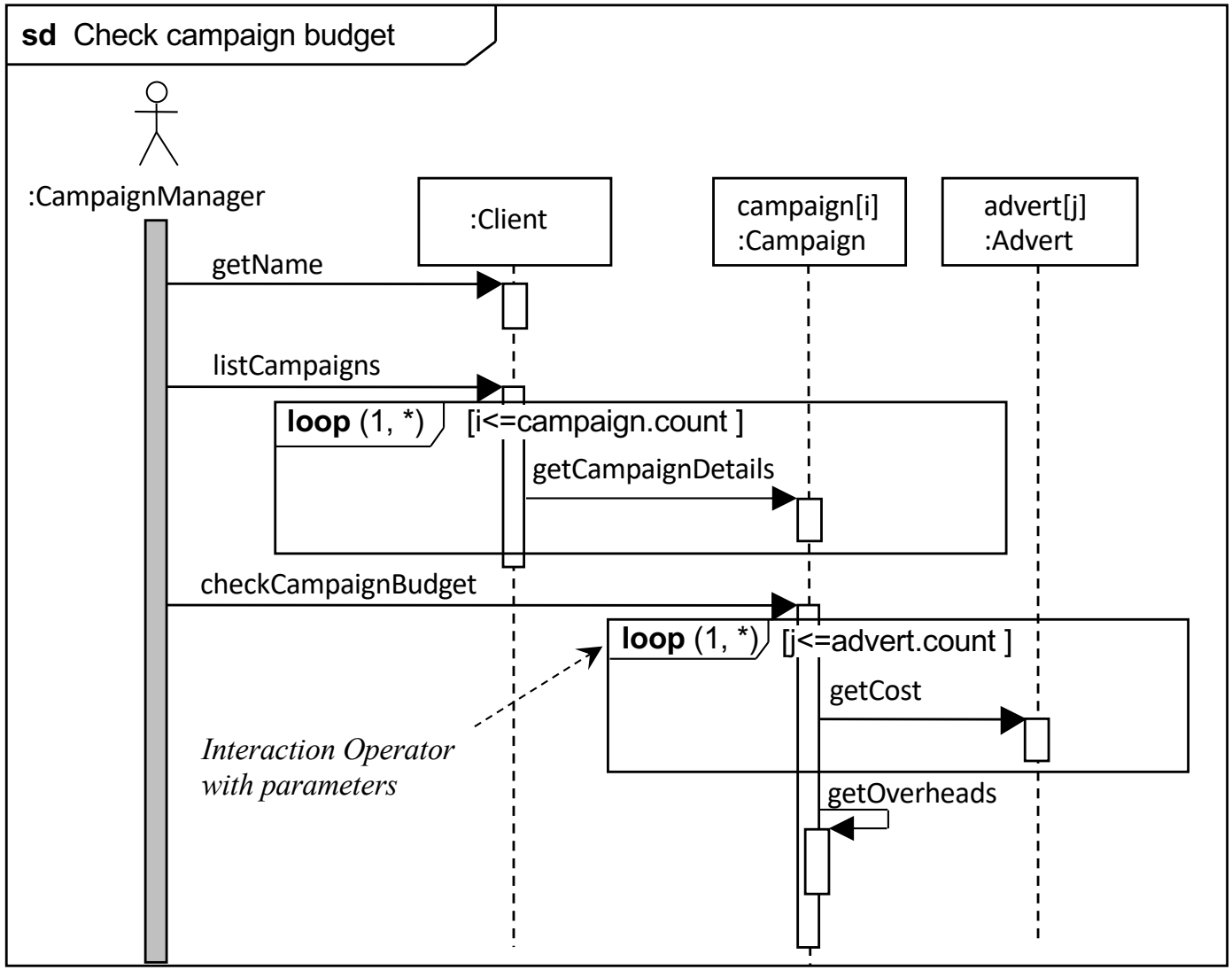
- Do not show logic flow

# Type diagrams



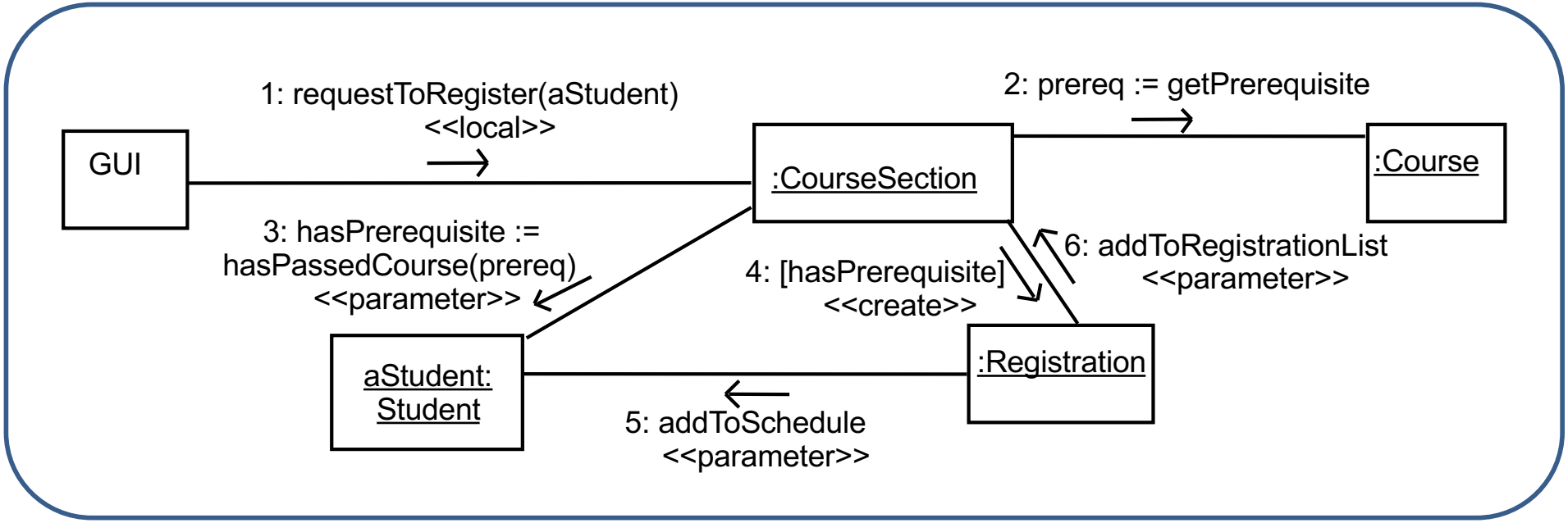

- Used during analysis to conceptually model a problem
- Based on requirements
- Only use if needed
- Essentially an ER diagram with added 'Types'
- Not part of UML
- You may not need to use it!

# Sequence diagrams



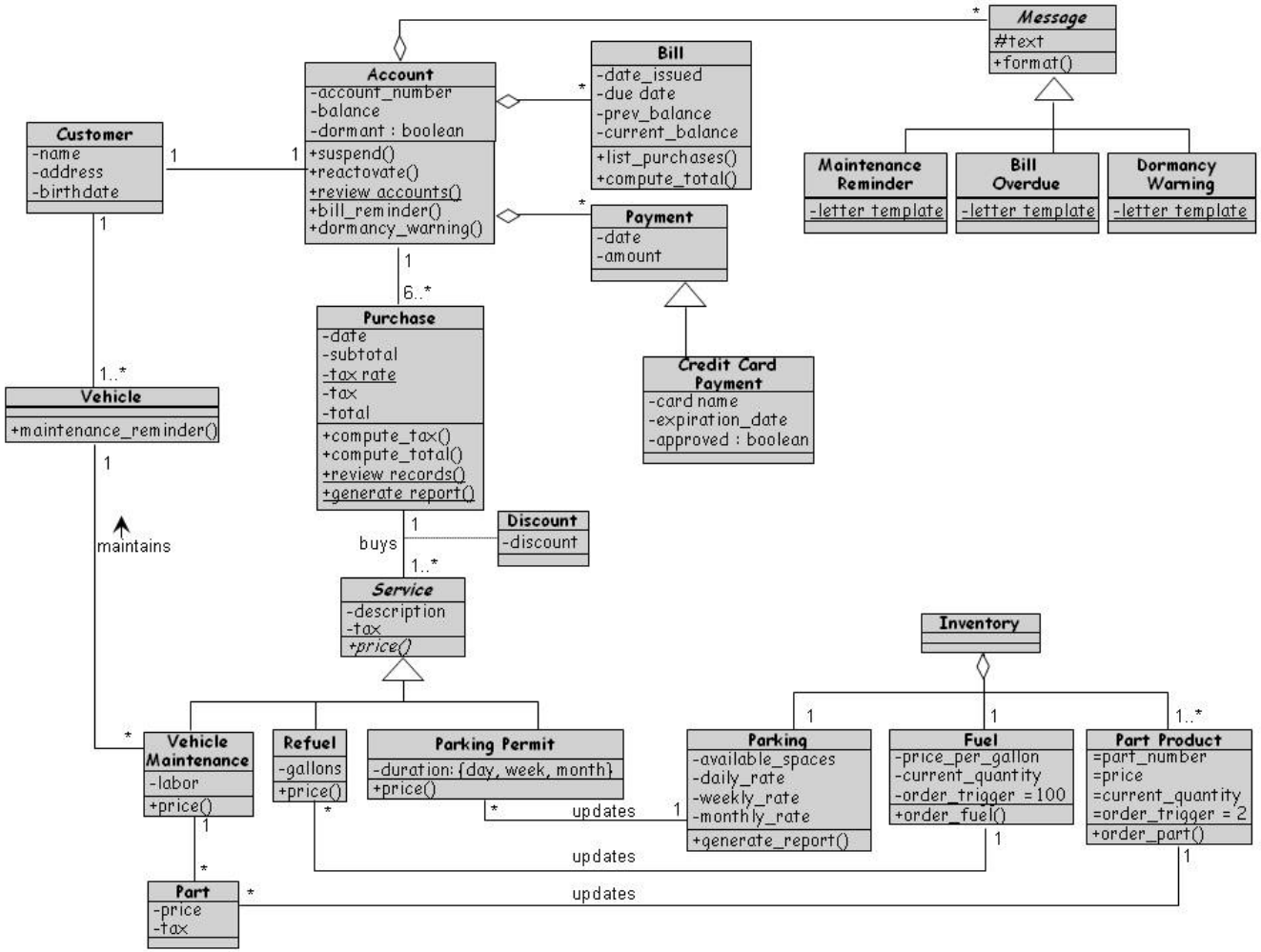

- Based on **use case descriptions**
- Models the flow of messages between objects involved in a use case
- Often provide a bridge between use cases and the class diagram
- Need to ensure consistency with **class diagram** (objects, associations and operations), and any **state machines**
- Beforehand you will need to decide on your **candidate classes/objects**

# Collaboration diagrams




- An alternative to the sequence diagram
- Shows exactly the same information (just numbers messages instead to show the order)
- Governed by the same rules as the sequence diagram
- Same consistency checking as sequence diagrams
- Can be easier to map to a class diagram
- Both can be used to generate template code for your classes

# Class diagram




Associations between CLASSES

- Named association with cardinality and direction (can be recursive, can have association classes)
- Inheritance or
- Whole/Part relationships:
  - Aggregation or
  - Composition or

  (can be included in a named association)

- The most important diagram
- Bridges between analysis and design
- Based on:
  - Analysis of requirements
  - Interaction diagrams
  - Needs to be consistent
- Check consistency with **interaction diagrams** and **state machines**

# Relationships in diagrams

**Use-case diagrams**

Between USE CASES

- <<extend>>
- <<include>>
- <<generalise>> or △

**Class diagrams**

Between CLASSES

- Named association with cardinality and direction (can be recursive, can have association classes)
- Inheritance or △
- Whole/Part relationships:
  - Aggregation or ◇
  - Composition or ◆

  (can be included in a named association)

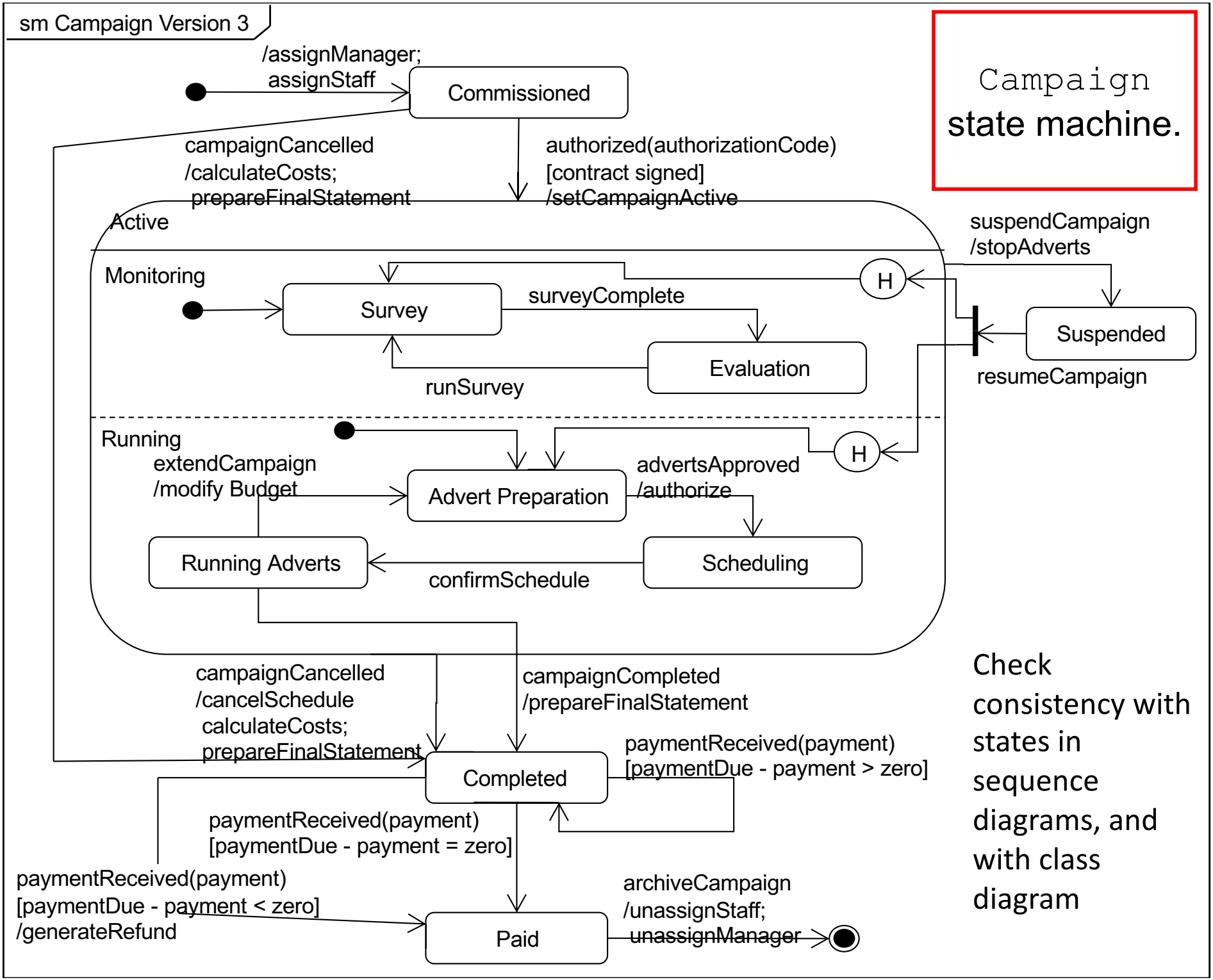
# CRC cards

| Class Name | *Client* |
|---|---|
| Responsibilities | Collaborations |
| *Provide client information.*<br>*Provide list of campaigns.* | *Campaign provides campaign details.* |

| Class Name | *Campaign* |
|---|---|
| Responsibilities | Collaborations |
| *Provide campaign information.*<br>*Provide list of adverts.*<br>*Add a new advert.* | *Advert provides advert details.*<br>*Advert constructs new object.* |

| Class Name | *Advert* |
|---|---|
| Responsibilities | Collaborations |
| *Provide advert details.*<br>*Construct adverts.* | |

- One CRC card per Class
- Responsibilities will need to be supported by the Class **attributes** and **operations**.
- Collaborations must match with Class diagram **associations**.




Check consistency with states in sequence diagrams, and with class diagram

# Consistency checking your state machine

- Every **event** should appear as an incoming message for the appropriate object on an **interaction diagram**(s).
- Every **action** should correspond to the execution of an **operation** on the appropriate **class**, and perhaps also to the dispatch of a message to another object.
- Every **event** should correspond to an **operation** on the appropriate **class** (but note that not all operations correspond to events).
- Every **outgoing message** sent from a state machine must correspond to an **operation** on another **class**.

# Analysis Flow in UML (reminder)

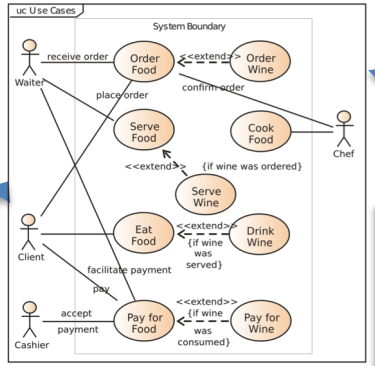
1

## Requirements

Problem Definition
**Viewpoints/Scope**
Functional & Non-Functional requirements
Scenarios

2

## Use-case diagrams




Actors

A sequence diagram for each use-case description

Names in bubbles match with use-case description names

3

## Use-case descriptions

| | |
|---|---|
| **Name** | The Use Case name. Typ[illegible]ame is of the format <action> + <object>. |
| **ID** | An identifier that is unique to [illegible] Use Case. |
| **Description** | A brief sentence that states wh[illegible] the user wants to be able to do and what benefit he will derive. |
| **Actors** | The type of user who interacts with the system to accomplish the task. Actors are identified by role name. |
| **Organizational Benefits** | The value the organization expects to receive from having the functionality described. Ideally this is a link directly to a Business Objective. |
| **Frequency of Use** | How often the Use Case is executed. |
| **Triggers** | Concrete actions made by the user within the system to start the Use Case. |
| **Preconditions** | Any states that the system must be in or conditions that must be met before the Use Case is started. |
| **Postconditions** | Any states that the system must be in or conditions that must be met after the Use Case is completed successfully. These will be met if the Main Course or any Alternate Courses are followed. Some Exceptions may result in failure to meet the Postconditions. |
| **Main Course** | The most common path of interactions between the user and the system.<br>1. Step 1<br>2. Step 2 |
| **Alternate Courses** | Alternate paths through the system.<br>AC1: <condition for the alternate to be called><br>1. Step 1<br>2. Step 2<br><br>AC2: <condition for the alternate to be called><br>1. Step 1 |
| **Exceptions** | Exception handling by the system.<br>EX1: <condition for the exception to be called><br>1. Step 1<br>2. Step 2<br><br>EX2 <condition for the exception to be called><br>1. Step 1 |

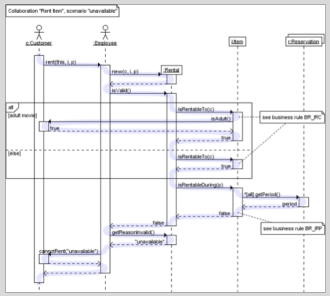
4

**List of Candidate Classes:**
Campaign
Order
Customer

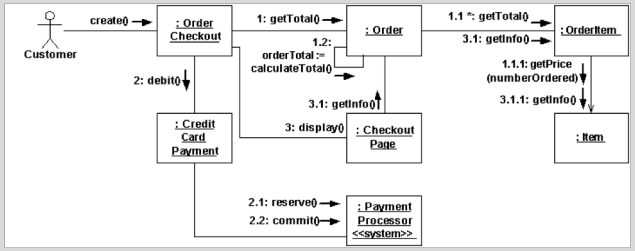
Class names

5

## Sequence diagrams



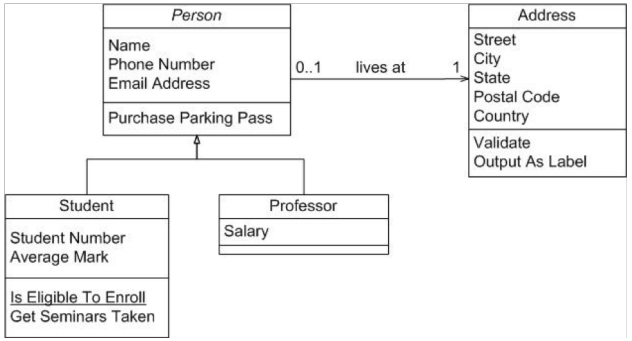
## Collaboration diagrams




Class Names
Messages/Associations

6

## Class diagram



# End

mgardner@essex.ac.uk