# 260CT
# Software Engineering

**Dr. Yih-Ling Hedley**
**Email: aa0817@coventry.ac.uk**

## Requirements Engineering 1

- Requirements engineering (RE) - Sommerville

*Requirements engineering emphasizes the use of* ***systematic and repeatable*** *techniques that ensure the* ***completeness, consistency, and relevance of the system requirements***

- RE encompasses:
  - **requirements elicitation**
  - **requirements analysis**
  - **requirements specification**
  - **requirements verification**
  - **requirements management**

## Requirements Engineering 2

- Requirements elicitation is: the process of **discovering, reviewing, documenting, and understanding** the user's needs and constraints for the system.
- Requirements analysis: the process of **refining** the user's needs and constraints.
- Requirements specification : the process of documenting the user's needs and constraints **clearly and precisely**.
- Requirements verification: the process of ensuring that the system requirements are **complete, correct, consistent, and clear**.
- Requirements management is the process of **scheduling, coordinating, and documenting** the requirements engineering activities

Source: Dorfman, M. & Thayer, R. H. *Software Requirements Engineering*

## Users Requirements: Review 1

- Information collected from requirements gathering stage: three categories:
  - **Functional requirements**
  - **Non-functional requirements**
    - **Usability requirements**

## Users Requirements: Review 2

- Requirements: categories
  - **Functional requirements:** describe what a system will do, referred to as **functionality**
  - Including:
    - **Descriptions of the processing** to be carried out
    - Details of **inputs** into the system from: documents, interactions between people and from other systems
    - Details of **outputs** expected from the system, such as report and screen displays
    - Details of **data** to be recorded in the system

## Users Requirements: Review 3

- Requirements: categories
  - **Non-functional requirements:** describe aspects of system concerning how well the system provides functional requirements

## Activity 1

- Identify non-functional requirements.

## Activity 1: Feedback

- Examples of non-functional requirements: (but not limited to)
  - **Performance**
  - **Multi-user and concurrent accessibility**
  - **System availability** with minimum of downtime
  - **System recovery** time from failure
  - **Data volumes** in terms of transaction throughput and **storage**
  - **Security** (e.g. resistance to attacks)

## Users Requirements: Review 4

- Requirements: categories
  - **Usability requirements:** concerning the system developed meets the **needs of the user** and **tasks** to undertake
  - Usability of a system should achieve:
    - **Effectiveness**
    - **Efficiency**
    - **Satisfaction**

## Activity 2

- Identify a number of fact-finding techniques to gather user requirements.

## Activity 2: Feedback

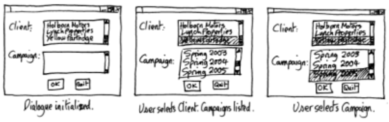
- Fact-finding techniques to gather user requirements:
  - **Background reading**
  - **Interviewing**
  - **Observation**
  - **Document sampling**
  - **Questionnaires**

## Requirements Modelling:
### Requirements Documentation

- **Documenting Requirements: a mixture of diagrams, data and text,** may include:
  - UML models
  - Records of interviews and observations
  - Details of problems, requirements and users
  - Meetings minutes
  - Copies of existing documents

## Requirements Modelling: Prototyping

- **Prototyping:**
  - Supports use case modelling
  - Help elicit requirements
  - Techniques:
    - **User interface prototypes  using tools such as** such as visual programming
    - **Storyboard in paper sketches**



## Requirements Modelling to Requirements Analysis

- Progressing from **Requirements Model to Requirements Analysis**
  - **Requirements Modelling**: requirements documentation, use case diagram, interface prototypes
  - **Requirements Analysis:** analysis class diagram, operation specifications

## Requirements Analysis: Operation Specifications

- **Requirements Analysis: Operation specifications**
  - **Analysis view point**: enables analyst to meet users' requirements
  - **Design view point**: provides a basis for detailed design specifications

## Operation Specifications: Algorithmic

- **Algorithmic** Approach
  - gives sequence of the steps; no need to concern about efficiency at analysis stage
    - **Programming control structures**
    - **Structured English**
    - **Pseudo-code**: uses syntax of a specific programming language
    - **UML Activity diagrams**: actions provide steps in operation logic

## Operation Specifications: Non-Algorithmic

- **Non-Algorithmic** Approach
  - describes an operation logic as a black box
    - **Pre- and post-conditions:**
      - ✓What conditions must be satisfied **before** an operation can take place?
      - ✓What conditions and states may the system be in **after** an operation is completed?
    - **Decision tables**: a matrix which shows the conditions under a decision is made, resultant actions and their relationship

## Operation Specifications: Pre- and post-conditions

- **Non-Algorithmic** Approach
  - Pre- and post-conditions
  - Example: Advert.getCost(): Money

Advert.getCost(): Money
**pre-conditions:** none
**post-conditions:** a valid Money value is returned

© Bennett, McRobb and Farmer 2010

## Operation Specifications: Decision tables

Conditions to be tested

rules

| Conditions and actions | Rule 1 | Rule 2 | Rule 3 |
|---|---|---|---|
| **Conditions** | | | |
| Is budget likely to be overspent? | N | Y | Y |
| Is overspend likely to exceed 2%? | - | N | Y |
| **Actions** | | | |
| No action | X | | |
| Send letter | | X | X |
| Set up meeting | | | X |

Action occurs if conditions are true

Possible actions

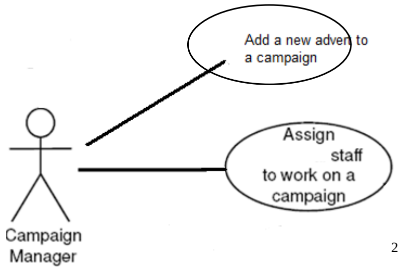
© Bennett, McRobb and Farmer 2010

## Analysis Model: Use Case Realisation

- Use Case Realisation:
  - each use case is realised by a series of models towards the implementation of the software that meets the requirements identified by that use case
  - identify a possible set of **classes and their interaction** to deliver the **functionality** of a use case
  - to produce an analysis model by:
    - developing **separate class diagrams for each use case**, then
    - **combining separate use case class diagrams into one** analysis class model

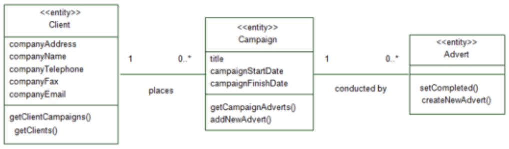
Source: © Bennett, McRobb and Farmer 2005

## Use Case Realisation:
### Use Case Examples: Assign staff to work on a campaign, Add a new advert to campaign


Add a new advert to a campaign
Assign staff to work on a campaign
Campaign Manager

## Use Case Realisation: Analysis class diagram
### Stage 1: Create Class Diagrams from Individual Use Cases

- Use Case Example: Add a new advert to campaign



Source: © Bennett, McRobb and Farmer 2005

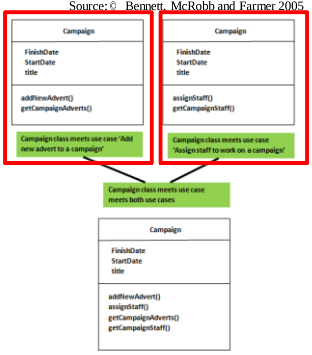
## Requirements Analysis
### Stage 2: Assembling Analysis Class Diagram 1

- from use case realisation (via different use cases) to one single analysis diagram, which consists of:
  - A single package of **entity classes**
    - **Assembles all attributes and operations into a single class definition**, as each entity class may be defined in different ways through various use cases

## Stage 2: Assembling Analysis Class Diagram 2

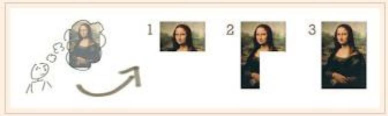
Source: © Bennett, McRobb and Farmer 2005

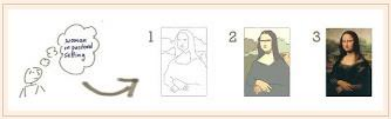
- **Example**: to assembles all attributes of individual classes (e.g. ***Campaign)*** from each use case realisation into a single class diagram



# Activity 3

- Identify the correct presentation for the two development processes: **Iteration and Incrementation**





# Activity 3: Feedback

- An **iteration** represents **the state of the overall development** and the **complete deliverable system**.

- An **increment** represents **the current work in progress** that will be **combined with the preceding iteration to form the next iteration and** a **shippable feature to add on**.

Source: by Jeff Patton