Software Design

- 'Requirements' defines
  - The goals the system needs to satisfy.
- 'Specification' defines
  - The externally-observable behaviour of the system.
- 'Architecture' defines
  - The major system-level components
  - Their methods of interaction
  - Technology used
- 'Design' of a component
  - The code that needs to be written.
  - We will focus exclusively on OO design.

Output of Design

- A document:
  - Prose description
  - UML
    - Classes
      - Associations
      - Methods
      - Attributes
    - Object diagrams
    - Sequence and collaboration diagrams
    - Statechart and activity diagrams
  - Formulae & algorithms
- Must relate to the architecture
- Can reference the requirements and specification
- Must be sufficient to allow coding to commence

Necessary Background

- Experience in OO programming
- Experience in OO analysis
- An understanding of OO concepts
  - Encapsulation (data hiding)
  - Objects, classes, meta-classes
  - Classes v.s. interfaces (types)
  - Inheritance, multiple inheritance
  - Polymorphism (run-time typing)
  - Implementation inheritance v.s. interface inheritance

Goal

- To teach you to create good OO designs.
- What you need:
  - Tools (UML notation)
  - Experience
- How we will teach you:
  - UML in tutorials
  - Next best thing to experience:
    - Other people's experience
    - Design Patterns

Object-Oriented Design

- Take OOA and extend to build the 'Problem Domain Component'
- Design the 'Human Interaction Component'
- Design the 'Task Management Component'
- Design the 'Data Management Component'

[Peter Coad & Edward Yourdon, *OOD*, Yourdon Press, 1991]

Problem Domain Component

- Reuse design and programming classes.
- Group problem-domain-specific classes and establish a protocol by adding generalization classes.
- Accommodate inheritance limitations in implementation language.
- Add design classes
  - Associations
  - Run-time modifiability
  - …
- Improve performance
  - Speed, memory, perceived speed
- Support the data management component