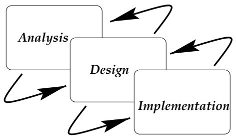
Object-Oriented Design

Derek G. Bridge

Room 336, Science Building
d.bridge@cs.ucc.ie
http://www.cs.ucc.ie/dbridge.html

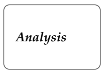
- Course web page:
  `http://www.cs.ucc.ie/~dgb/courses/ood.html`
- Lectures
- Private study
- Written exam ($1\frac{1}{2}$ hours, worth 80%)
- Year's work (Spring term team project, worth 20%)

What is Design? What is a Good Design?

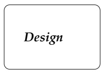


Consider an analogy with building construction

Consider the models that get built at each step



- Identify a problem
- Analyse the problem
- Specify requirements of the solution
  - functional requirements
  - non-functional requirements
- Be aware of all stake-holders, who may have requirements



- Design as
  - invention (to formulate alternatives)
  - model-building (to describe the alternatives)
  - decision-making (to rank & choose between alternatives)
- High-level design (architecture) and detailed design
- The myth of stable requirements
  - everybody's understanding improves as ever more concrete/detailed models get built

Iteration

- Systems should be built in iterations, each iteration being a mini-project concluding with a demo to 'someone'
- Each iteration adds functionality
- Each iteration will involve rewriting some existing code (read up on Refactoring)
- The process should be risk-driven: tackling riskiest parts first reduces overall risk of failure
- Iterative development increases planning & control!

Abstraction = Encapsulation + Information Hiding

- An abstraction highlights relevant properties and suppresses irrelevant details.
- Abstraction is essential for understanding large, complex systems.
- Encapsulation means the grouping together of elements to form larger elements.
- Information Hiding means concealing details within elements.
- If we form the right elements and, within these, hide the right details, we may get good abstractions that aid understandability.
- Software development is a process of suppressing less and less detail.

Abstraction quality

- Many different 'groupings of elements' in software
  - group statements into blocks
  - group statements and blocks into methods/ procedures/ functions
  - group variable declarations and methods into class definitions
  - group class definitions into class hierarchies
  - group class definitions into packages
  - (others: group class definitions into files; nest procedures within one another; nest class definitions within one another)

Cohesion

> *The cohesion of a group is the degree to which the elements in the group are directed at a single purpose.*

High cohesion is best.

Highly cohesive groups are

- more easily understood
- more easily re-used
- more easily maintained
- etc.

Coupling

> *Coupling is the degree to which groups depend on each other.*

Low coupling is best.

If there are minimal dependencies between groups, then the groups are

- more easily understood
- more easily re-used
- more easily maintained
- etc.

Case study: coupling between methods

Consider two methods (procedures/ functions), A and B.

Give examples of how they could be coupled (from low to high coupling)