Inf2-SEPP
Lecture 9 Part 1: Detailed design. Software design principles

Cristina Adriana Alexandru

School of Informatics
University of Edinburgh

Previous lecture

- Design
  - Concept
  - Outputs of the design process
  - Criteria for good design
  - Levels of design
    - 1. Architectural design

This lecture

- Levels of design
  - 2. Detailed design
- Software design principles
  - Cohesion
  - Coupling
  - Abstraction
  - Encapsulation/information hiding
  - Separation of interface and implementation
  - Decomposition, modularisation

Detailed design

Happens inside a subsystem or component.

E.g.:

- System architecture has been settled by a small team written down, and reviewed.
- You are in charge of the detailed design of one subsystem.
- You know what external interfaces you have to work to and what you have to provide.
- Your job is to choose classes and their behaviour that will do that.

Idea: even if you're part of a huge project, your task is now no more difficult than if you were designing a small system.

But: your interfaces may seem artificial, which makes them harder to understand and adhere to.

Software Design Principles

Key notions that provide the basis for many different software design approaches and concepts.

Design Principles: initial example

Which of these two designs is better?

A)
```
public class AddressBook {
  private LinkedList<Address> theAddresses;
  public void add (Address a) {theAddresses.add(a);}

  // ... etc. ...
}
```

B)
```
public class AddressBook extends LinkedList<Address> {
  // no need to write an add method, we inherit it
}
```

C) Both are fine

D) I don't know

Design Principles: initial example (cont.)

A is preferred.

- an `AddressBook` is not conceptually a `LinkedList`, so it shouldn't extend it.
- If B chosen, it is much harder to change implementation, e.g. to a more efficient `HashMap` keyed on name.

Design principles 1

Cohesion is a measure of the strength of the relationship between pieces of functionality within a component.

High cohesion is desirable.

Benefits of high cohesion include increased understandability, maintainability and reliability.

Design principles 2

Coupling is a measure of the strength of the inter-connections between components.

Low or loose coupling is desirable.

Benefits of loose coupling include increased understandability and maintainability.

Design principles 3

- abstraction - procedural/functional, data
  The creation of a view of some entity that focuses on the information relevant to a particular purpose and ignores the remainder of the information
  e.g. the creation of a sorting procedure or a class for points

- encapsulation / information hiding
  Grouping and packaging the elements and internal details of an abstraction and making those details inaccessible

- separation of interface and implementation
  Specifying a public interface, known to the clients, separate from the details of how the component is realized.

Design principles 4

- decomposition, modularisation
  dividing a large system into smaller components with distinct responsibilities and well-defined interfaces

Reading

Essential: Stevens Chapter 1 section 1.3

Recommended: return to any mentions of cohesion, coupling, abstraction, encapsulation, separation of interface and implementation, decomposition from your Inf1B course.

Recommended: SWEBOK v3 Ch2 for an overview of the field of software design