SED Lecture 11
Design Patterns

Objectives (also continued in lecture 14)
To understand the main design patterns concepts and their application to software design: modifiability; reusability; and a unified vocabulary. To understand the key concepts of Object Oriented Software Design: Abstraction; Encapsulation; and Polymorphism, in relation to design patterns. To explain the design patterns: Observer; Abstract Factory; Adapter; Bridge and Façade. Given a standard description, be able to interpret a design pattern.

Key Points
- Design Patterns are template solutions to recurring design problems. They are less abstract than Software Architectures (e.g. a layered architecture), though more abstract than data structures (e.g. a linked list).
- Design Patterns exploit Object Oriented concepts, such as polymorphism and encapsulation to produce reusable and modifiable solutions to common design problems.
- Designs Patterns are catalogued, and each pattern has four essential elements: a name; a description of the problem context; a description of the elements that make up the solution; and the consequences of applying the pattern.
- Reusable classes allow a verified solution to be applied in many different contexts, and greatly improve software reliability.
- Modifiable classes allow components to vary according to the changing requirements of a system and allow software problems to be fixed with little risk of new problems emerging.
- In these two lectures we will examine the following design patterns: Observer; Singleton; Abstract Factory; Adapter; Bridge and Façade.
- The Observer pattern defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.
- The Singleton defines a static operation that lets clients access its unique instance.
- The Abstract Factory pattern provides an interface for creating families of related or dependent objects without specifying their concrete classes.
- The Adapter pattern converts the interface of a class into another interface that clients expect.
- The Bridge pattern decouples an abstraction from its implementations so the two can vary independently.
- The Façade pattern provides a unified interface to a set of interfaces in a subsystem.

Core reading
Brugge & Dututoit, Chapter 8 and appendix A: design patterns

Gamma, Helm Johnson Vlissides, Design Patterns: Elements of Reusable Object Oriented Software, Addison-Wesley, Reading, MA, 1994

UWA CITS 2220, Semester , 2011