What these slides will cover:

• What is a pattern?
• What types of pattern are there?
• Why do we use patterns in software architecture?
• What does a pattern look like?
• How can we use patterns in our work?

What is a pattern?

Definition:
A particular recurring design problem that arises in specific design contexts, and presents a well-proven generic scheme for its solution. The solution scheme is specified by describing its constituent components, their responsibilities and relationships, and the ways in which they collaborate.

Taken from Pattern-Oriented Software Architecture, Buschmann et al.
Definition in English...

- A re-usable solution to a recurring problem
- Tried and tested
- Consider the solution to be a template
- It can be adapted and personalised for the problem domain

Pattern categories

- 3 categories of patterns defined by Buschmann et al.
- Architectural patterns
- Design patterns
- Idioms
- But there’s more...
- Analysis patterns (Martin Fowler)
- Organisational patterns

Three categories of patterns

**Architectural Patterns**
- A high-level structure for software systems
- Contains a set of predefined sub-systems
- Defines the responsibilities of each sub-system
- Details the relationships between sub-systems
- Also similar to ‘conceptual patterns’ which cover the application domain (defined in *Understanding and Using Patterns in Software Development*, Riehle & Zullighoven)
Three categories of patterns (cont)

Design Patterns
- Mid-level construct
- Implementation-independent
  - Designed for ‘micro-architectures’ – somewhere between sub-system and individual components

- Several classic design patterns described in Design patterns: elements of reusable object-oriented software, Erich Gamma et al.

Three categories of patterns (cont)

Idioms
- Earliest form of software pattern
- Comparatively low-level
- Gives a guide for implementing the components and relationships of the pattern
- Considers the pattern at a programming language level
  - Describes the pattern using the constructs of the specific language

- Also similar to ‘programming patterns’ (Riehle & Zühlighoven again)

Pattern format
- A pattern description should contain the following elements:
  - Name
  - Problem
  - Context
  - Forces
  - Solution
  - Examples
  - Resulting context
  - Rationale
  - Related patterns
  - Known uses

- A pictorial representation may also be included, as may an abstract
Pattern elements

- Name
  - Meaningful, concise

- Problem
  - A description of intent: goals and objectives of the pattern

- Context
  - The preconditions of the problem and solution
  - Where the pattern is applicable

- Forces
  - Motivations and trade-offs to be made in the design and implementation; may be conflicting
  - For example: maintainability, security, efficiency…

Pattern elements (cont)

- Solution
  - Consists of static relationships and dynamic rules
  - Described by pictures, diagrams, text
  - Contains implementation guidelines (and what to avoid doing)

- Examples
  - To help the user understand its application more fully
  - The consequences of applying the pattern
  - Resolves which forces have been addressed

- Rationale
  - A justification of how and why the pattern works

Pattern elements (cont)

- Related patterns

- Known uses
**Useful references**

- **Books:**
  - *Pattern-oriented Software Architecture: System of Patterns* – Frank Buschmann et. al
  - *Design patterns: elements of reusable object-oriented software* – Erich Gamma et. al

- **Online:**
  - [http://g.oswego.edu/dl/pd-FAQ/pd-FAQ.html](http://g.oswego.edu/dl/pd-FAQ/pd-FAQ.html)