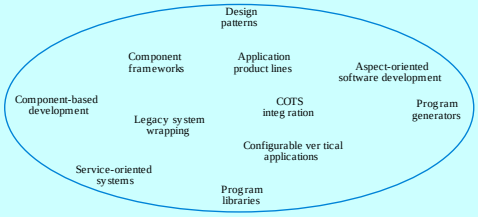
Design Patterns

ECSC409: Software Engineering Principles

1

The reuse landscape



Reuse approaches 1

Design patterns	**Generic abstractions that occur across applications are represented as design patterns that show abstract and concrete objects and interactions.**
Component-based development	Systems are developed by integrating components (collections of objects) that conform to component-model standards.
Application frameworks	Collections of abstract and concrete classes that can be adapted and extended to create application systems.
Legacy system wrapping	Legacy systems that can be 'wrapped' by defining a set of interfaces and providing access to these legacy systems through these interfaces.
Service-oriented systems	Systems are developed by linking shared services that may be externally provided.

Reuse approaches 2

Application product lines	An application type is generalised around a common architecture so that it can be adapted in different ways for different customers.
COTS integration	Systems are developed by integrating existing application systems.
Configurable vertical applications	A generic system is designed so that it can be configured to the needs of specific system customers.
Program libraries	Class and function libraries implementing commonly-used abstractions are available for reuse.
Program generators	A generator system embeds knowledge of a particular types of application and can generate systems or system fragments in that domain.
Aspect-oriented software development	Shared components are woven into an application at different places when the program is compiled.

The concept *reuse*

- When you reuse program or design components, you have to follow the design decisions made by the original developer of the component.
- This may limit the opportunities for reuse.
- However, a more abstract form of reuse is concept reuse when a particular approach is described in an implementation independent way and an implementation is then developed.
- The two main approaches to concept reuse are:
  - **Design patterns;**
  - **Generative programming.**

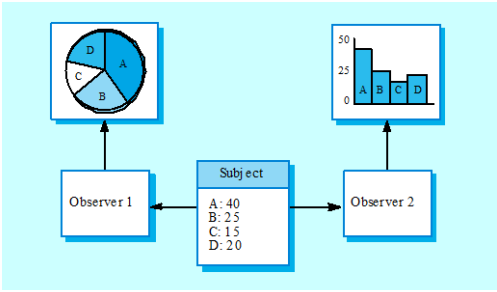
Design patterns

- A design pattern is a way of reusing abstract knowledge about a problem and its solution.
- A pattern is a description of the problem and the essence of its solution.
- It should be sufficiently abstract to be reused in different settings.
- Patterns often rely on object characteristics such as inheritance and polymorphism.

Pattern elements

- Name
  - A meaningful pattern identifier.
- Problem description.
- Solution description.
  - Not a concrete design but a template for a design solution that can be instantiated in different ways.
- Consequences
  - The results and trade-offs of applying the pattern.

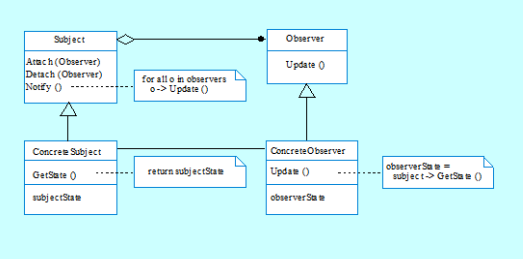
Multiple displays



The Observer pattern

- Name
  - Observer.
- Description
  - Separates the display of object state from the object itself.
- Problem description
  - Used when multiple displays of state are needed.
- Solution description
  - See slide with UML description.
- Consequences
  - Optimisations to enhance display performance are impractical.

The Observer pattern



Example Categories

- **Creational Patterns:**
  - E.g. Abstract Factory, Factory Method
- **Structural Patterns:**
  - ***Composite***
  - Proxy
- **Behavioural Patterns:**
  - E.g. Command, Visitor
- *These are from Gamma et al. (1995) (a.k.a. the Gang of Four book), but there are many other pattern collections.*

11

AnagramGame Example

- Can you see any design pattern?

12

A video clip about the Factory Method Design Pattern

- http://www.youtube.com/watch?v=RTCPwUCGn9A

13

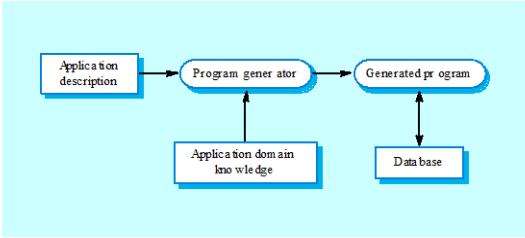
Generator-based reuse

- Program generators involve the reuse of standard patterns and algorithms.
- These are embedded in the generator and parameterised by user commands. A program is then automatically generated.
- Generator-based reuse is possible when domain abstractions and their mapping to executable code can be identified.
- A domain specific language is used to compose and control these abstractions.

Types of program generator

- Types of program generator
  - Application generators for business data processing;
  - Parser and lexical analyser generators for language processing;
  - Code generators in CASE tools.
- Generator-based reuse is very cost-effective but its applicability is limited to a relatively small number of application domains.
- It is easier for end-users to develop programs using generators compared to other component-based approaches to reuse.

Reuse through program generation



More on Design Patterns

- http://en.wikipedia.org/wiki/Design_pattern_(computer_science)

17

More on Patterns in Nature

- http://www.youtube.com/watch?v=SDrvzUitPoE

18

References from the on-line available literature

- Software Engineering Principles and Practice, Hans van Vliet, John Wiley & Sons 2008, Chapter 17 (Software Reusability)
- Software engineering for students, Doug Bell 1944-, 4th ed. Harlow, England ; New York : Addison-Wesley 2005, Part B, Section 12
- Software engineering, Ian Sommerville 1951-, 8th ed. Harlow, England ; New York : Addison-Wesley 2007, Chapter 18 (Software Reuse)

19

Further references

- Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software. Reading, MA: Addison-Wesley.
- T. Winn, P. Calder. Is This a Pattern?. IEEE Software, 19(1):59-66, January/February, 2002.
- F. Buschmann, K. Henney, D.C. Schmidt. Past, Present, and Future Trends in Software Patterns. IEEE Software, 24(4):31-37, July, 2007.

20