UECS2344 Software Design: Lecture 11

Reuse

Types of reuse: (from Object-Oriented Software Engineering (Lethbridge and Laganiere))

1. **Reuse of expertise**: Software developers who have many years of experience working on projects can save time when developing new systems because they do not need to re-think many issues: their past experience tells them what needs to be done. If such people write articles and books describing their experiences, this can help others to do better software engineering work.

2. **Reuse of standard designs and algorithms**: There are thousands of algorithms and other aspects of designs described in various books (e.g. in the form of design patterns and architectural patterns), standards documents, and articles. Software developers only need to implement them if they are appropriate to the current task.

3. **Reuse of libraries of classes or procedures or of powerful commands built into languages or operating systems**: Libraries and commands represent implemented algorithms, data structures and other facilities. Software developers commonly do this kind of reuse since all programming languages come with some basic libraries.

4. **Reuse of complete applications**: Software developers can take complete applications and add a small amount of extra software that makes the application behave in special ways the customer wants. For example, a software developer takes a standard email application and adds a feature that updates its 'address book' with data from the company's employee and client databases. This type of reuse is often called reuse of commercial off-the-shelf (COTS) software, and the extra code written is often called 'glue code'.

5. **Reuse of frameworks**. (see below)

6. **Reuse of components**. (see below)

Frameworks

A framework is reusable software that implements a generic solution to a generalised problem. It provides common facilities applicable to different application programs of a similar type.

The key principle behind frameworks is as follows: applications of a similar type tend to have similar features. For example, all graphical user interface (GUI) applications have buttons, text boxes, etc.

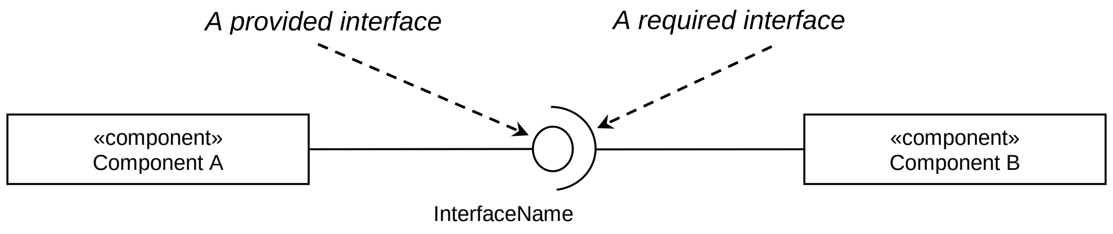
A framework will contain those common features. But it is not a complete application i.e. it is has missing parts. The developer fills in those missing parts based on the requirements of the specific application. Some of the missing parts are mandatory i.e. must be filled and some are optional.

A framework enables reuse of both design and code. The developer not only reuses the overall design envisioned by the framework designer, but also the code that implements that design.

Components

A software component is similar to a module but it contains executable code rather than source code. A component can provide services to other components and a component may require services from other components.

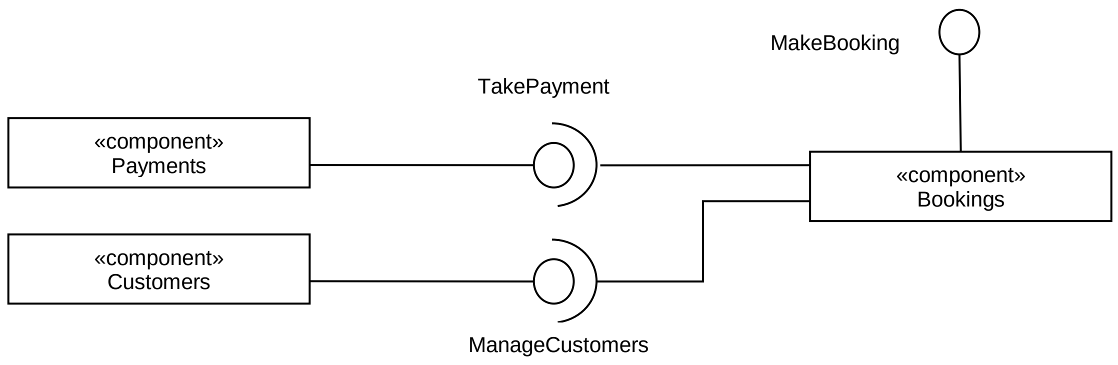
Component Diagram



This diagram shows that a provided interface of one component is mapped to a required interface of another component

A component has interfaces. An interface lists a range of services (or operations). An interface may be either a provided interface (in a providing component) or a required interface (in a requesting component). The providing component provides the services listed in its providing interface and the requesting component requests for the services listed in its required interface.

Example: Component Diagram for an airline system



The `Booking` component:

- uses an interface provided by the `Payments` component which handles the processing of credit cards and any other types of payment.
- uses an interface provided by the `Customers` component to obtain and update details of the airline's customers.
- provides an interface called `MakeBooking` which is available to the system used by travel agents and the e-commerce system on airline's website.

A component enables reuse of both design and code. Developers who require the services provided by a component may use the component without developing the component themselves.