CS281 - Software Engineering

Component-based Software Engineering

Topics covered

- ✧ Components and component models
- ✧ CBSE processes
- ✧ Component composition

Component-Based Software Development (CBSE)

- An approach to software development that relies on the reuse of entities called 'software components'.
- CBSE is based on sound design principles:
  - Components do not interfere with each other (i.e. independent)
  - Component implementations are hidden
  - Communication is through well-defined interfaces
  - Components can be replaced if the interfaces are maintained
  - Component infrastructures offer a range of standard services.

CBSE Essentials

- Independent components specified by their interfaces
- Component standards to facilitate component integration
- Middleware that provides support for component inter-operability
- A development process that is geared to reuse

Some of the component standards:

- Sun's Enterprise Java Beans
- Microsoft's COM and .NET
- CORBA's CCM

They are competing :(

Component as a service provider

- The component is an independent, executable entity. It does not have to be compiled before it is used with other components.
- The services offered by a component are made available through an interface and all component interactions take place through that interface.
- The component interface is expressed in terms of parameterized operations and its internal state is never exposed.

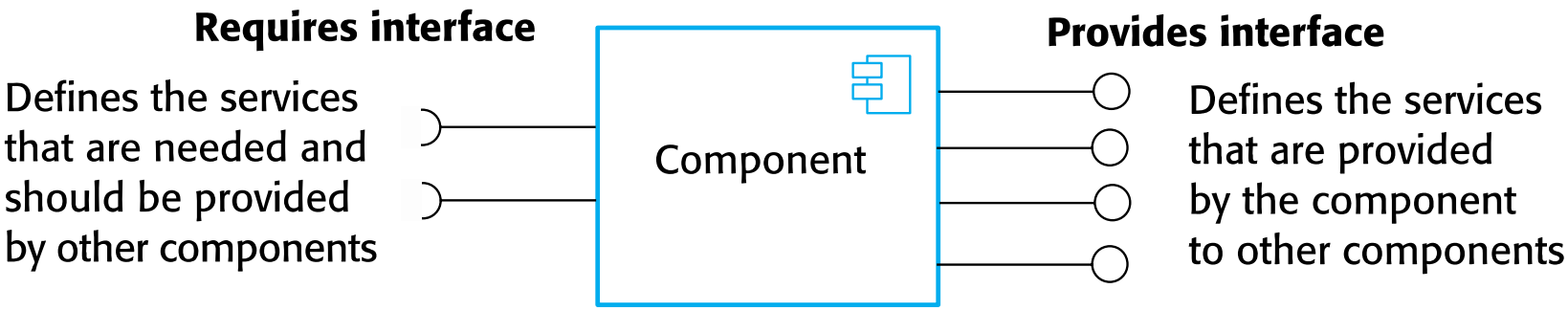
Component access

- Components are accessed using remote procedure calls (RPCs).
- Each component has a unique identifier (usually a URL) and can be referenced from any networked computer.
- Therefore it can be called in a similar way as a procedure or method running on a local computer.

Component interfaces

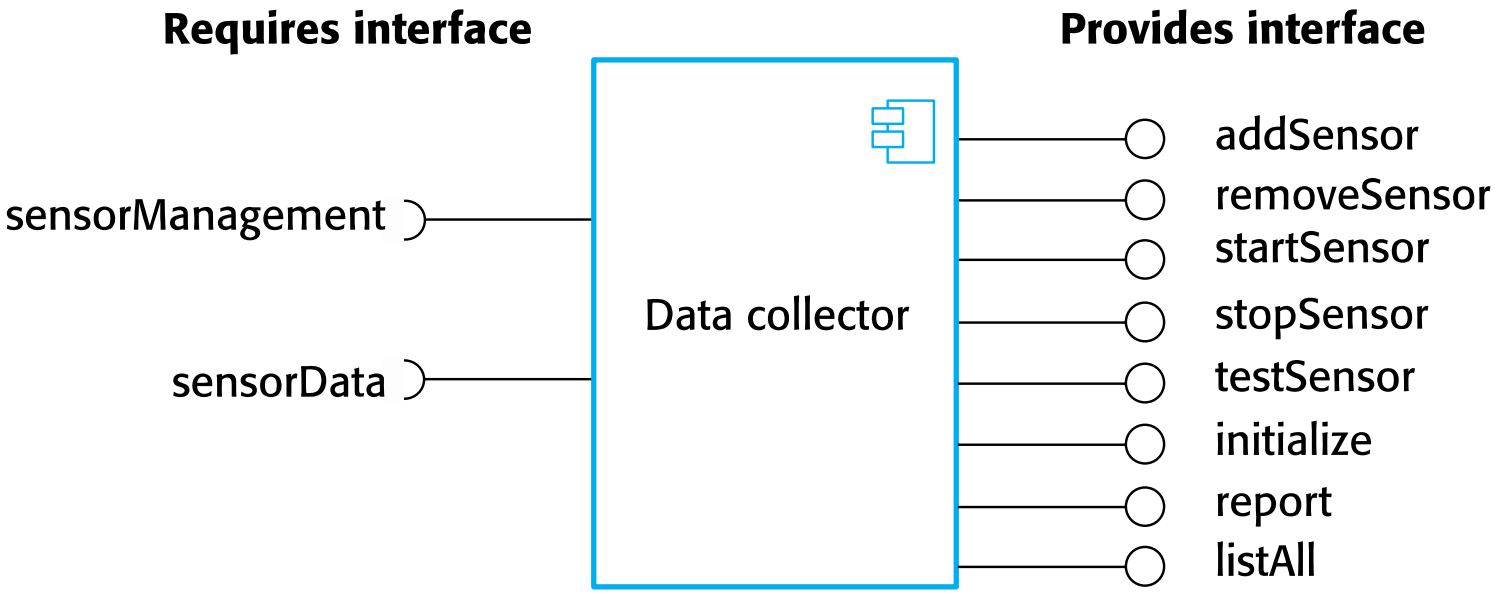
- ✧ Provides interface
  - Defines the services that are provided by the component to other components.
  - This interface, essentially, is the component API. It defines the methods that can be called by a user of the component.
- ✧ Requires interface
  - Defines the services that specifies what services must be made available for the component to execute as specified.
  - This does not compromise the independence or deployability of a component because the ‘requires’ interface does not define how these services should be provided.

Component interfaces



Note UML notation. Ball and sockets can fit together.

A model of a data collector component



Elements Of A Component Model

- A component model is a definition of standards for component implementation, documentation and deployment.
- Examples of component models:
  - EJB model (Enterprise Java Beans)
  - COM+ model (.NET model)
  - Corba Component Model
- The elements of a component model specifies:
  - Interfaces
  - Usage
  - Deployment

Middleware support

- Component models are the basis for middleware that provides support for executing components.
- Component model implementations provide:
  - Platform services that allow components written according to the model to communicate;
  - Support services that are application-independent services used by different components.
- To use services provided by a model, components are deployed in a container. This is a set of interfaces used to access the service implementations.

Legacy system components

- Existing legacy systems that fulfil a useful business function can be re-packaged as components for reuse.
- This involves writing a wrapper component that implements provides and requires interfaces then accesses the legacy system.
- Although costly, this can be much less expensive than rewriting the legacy system.

Reusable components

- The development cost of reusable components may be higher than the cost of specific equivalents. This extra reusability enhancement cost should be an organization rather than a project cost.
- Generic components may be less space-efficient and may have longer execution times than their specific equivalents.

Component identification issues

- Trust. You need to be able to trust the supplier of a component. At best, an untrusted component may not operate as advertised; at worst, it can breach your security.
- Requirements. Different groups of components will satisfy different requirements.
- Validation.
  - The component specification may not be detailed enough to allow comprehensive tests to be developed.
  - Components may have unwanted functionality. How can you test this will not interfere with your application?

Component validation

- Component validation involves developing a set of test cases for a component (or, possibly, extending test cases supplied with that component) and developing a test harness to run component tests.
  - The major problem with component validation is that the component specification may not be sufficiently detailed to allow you to develop a complete set of component tests.
- As well as testing that a component for reuse does what you require, you may also have to check that the component does not include any malicious code or functionality that you don't need.

Ariane launcher failure – validation failure?

In 1996, the 1st test flight of the Ariane 5 rocket ended in disaster when the launcher went out of control 37 seconds after take off. The problem was due to a reused component from a previous version of the launcher (the Inertial Navigation System) that failed because assumptions made when that component was developed did not hold for Ariane 5. The functionality that failed in this component was not required in Ariane 5!!

Component Composition

- The process of assembling components to create a system.
- Composition involves integrating components with each other and with the component infrastructure.
- Normally you have to write 'glue code' to integrate components.
- Glue Code: A code that allows components to work together. The glue code may be used also to resolve interface incompatibilities.

Interface Incompatibility

- Parameter incompatibility where operations have the same name but are of different types.
- Operation incompatibility where the names of operations in the composed interfaces are different.
- Operation incompleteness where the provides interface of one component is a subset of the requires interface of another.

Questions?