Component Models and Technology

Component-based Software Engineering

Ivica Crnkovic
ivica.crnkovic@mdh.se

Overview

- Introduction
- ACME Architectural Description Language
- Java Bean Component Model
- COM, DCOM, MTS and COM+
- CORBA Component Model (CCM)
- .NET Component Model
- OSGI Component Model

Architecture Definition Languages

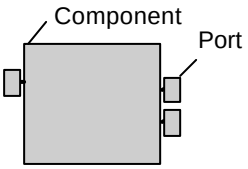
- **ADLs primarily address the issues related to the early phases of software engineering:**
  - Design
  - Analysis
- **They identify a number of concepts, such as:**
  - Architecture, configurations, connectors, bindings, properties, hierarchical models, style, static analysis and behavior.

ACME Architectural Description Language

- **Components and Ports**
- **Connectors and Roles**
- **Systems and Attachments**
- **Representations and Bindings**
- **Properties, Constraints, Types and Styles**

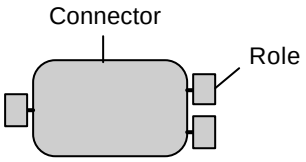
Components and Ports

- **Components**
  - Represent the computational elements and data stores of a system.
- **Ports**
  - Are the points of interaction between a component and its environment.



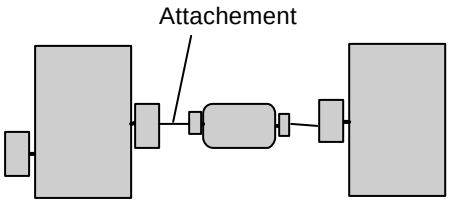
Connectors and Roles

- **Connectors**
  - Represent interactions between components such as method calls or an SQL connection between a client and a database server.
- **The interface of a connector is defined as a set of roles**

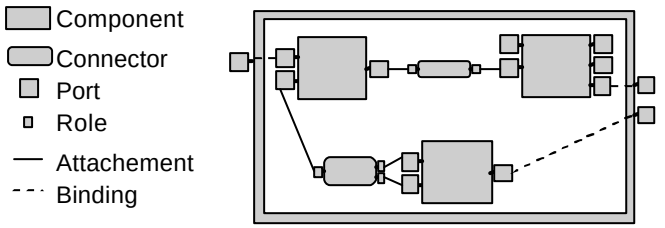


Systems and Attachments

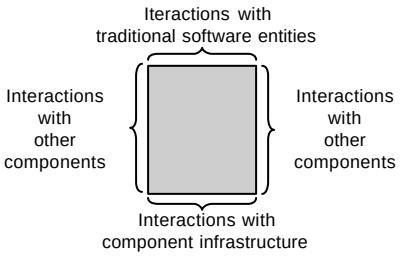
- **The structure of a system is specified by a set of components, a set of connectors, and a set of attachments.**
- **Attachment**
  - Links a component port to a connector role.

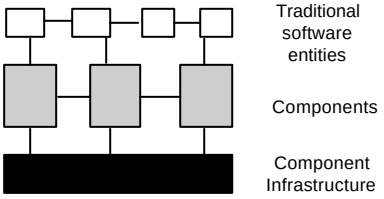


Representations and Bindings



Component Interactions





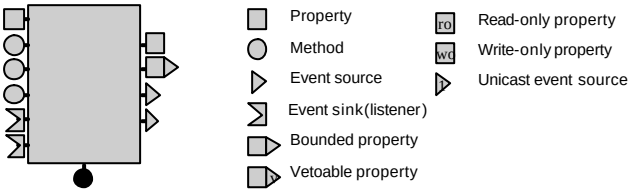
Java Bean Component Model

Key Features

- **"*A Java Bean is a reusable software component that can be manipulated visually in a builder tool*".**
- **The Java Bean was designed for the construction of graphical user interface (GUI).**
- **Explicitly tailored to interact in two different contexts:**
  - At composition time, within the builder tool.
  - At execution time, with the runtime environment.
- **Any Java class that adheres to certain conventions regarding property and event interface definitions can be a JavaBean.**
- **Beans are Java classes that can be manipulated in a visual builder tool and composed into applications.**

Interface of a Component

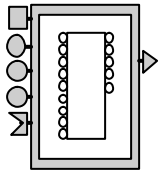
- **This model defines four types of port:**
  - methods,
  - properties,
  - event sources (generate an event)
  - event sinks called listeners (they receive event)

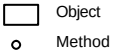


Ports

Implementation of a Component

- **Most bean components are implemented by a simple Java object by naming convention**

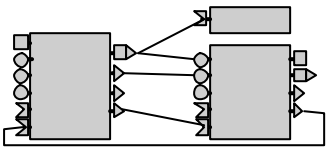




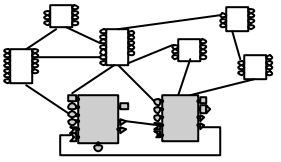
A simple implementation

Components Assembly

- **Assembly is one of the key features of Java Bean though no not specific solution is provided.**
  - Composition tools (Bean Box)
  - No composition language
- **Different ways of assembling components are supplied.**



Component-based assembly



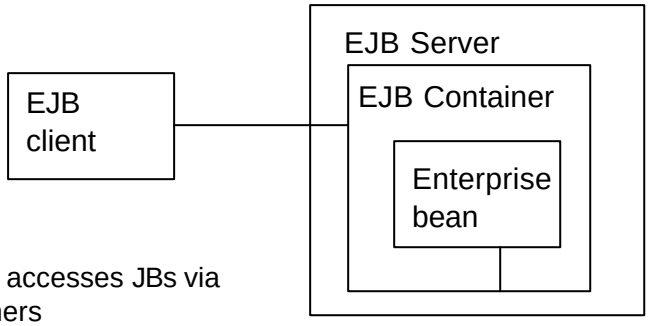
Heterogeneous assembly

Packaging and Deployment

- **Java Beans define a model for packaging components into archives.**
  - Includes the definition of dependency relationships between the package items.
- **Each package item can be marked "Design Only", so that they can be removed in a final application.**

Enterprise JavaBeans

- **Architecture for distributed applications**
- **Framework for creating middleware**



Clients accesses JBs via containers

EJB Servers

- **Analogous to CORBA ORB**
  - Naming and directory interface (JNDI)
  - Client finds EJB via JNDI
- **JTS**
  - Java Transaction Services

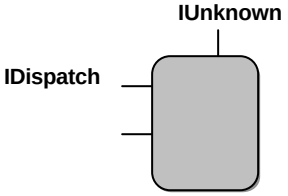
COM/DCOM

Interfaces and Assembly

- **A COM interface is seen as a C++ virtual class and takes the form of a list of data and function declarations without associated code.**
- **All interfaces are descendants of the IUnknown interface.**

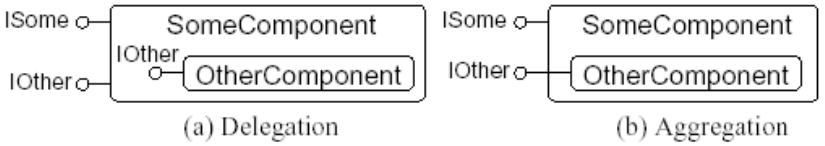
Implementation

- **COM defines a binary standard for interfaces**
  - Language independent implementation of the interfaces
- **Interfaces are defined in Interface Definition Language (IDL)**
- **An interface has an GUID as an identifier**



Composition

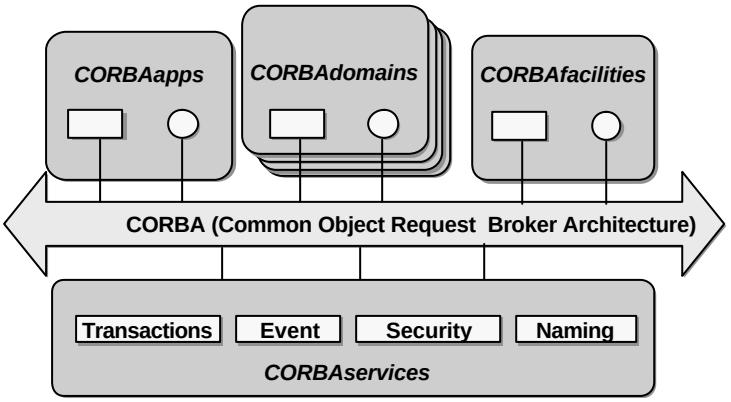
- **Two type of explicit composition**
  - Delegation
  - Aggregation
    - ☞For aggregation source code is needed



CORBA Component Model (CCM)

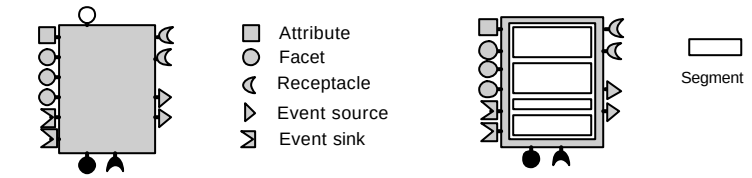
OMA Overview

OMA–Object Management Architecture



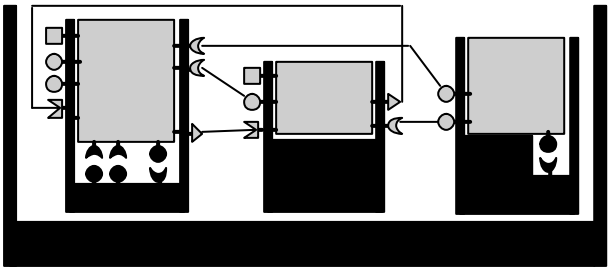
Interface and Assembly

- **A component interface is made of ports divided into:**
  - Facets - for interaction with clients
  - Receptacles – references o external code
  - Event sources
  - Event sinks



Framework : The Container Approach

- **Framework – a set of containers. Containers contains components and provides a set of standard services (security, events, persistence, life-cycle support)**

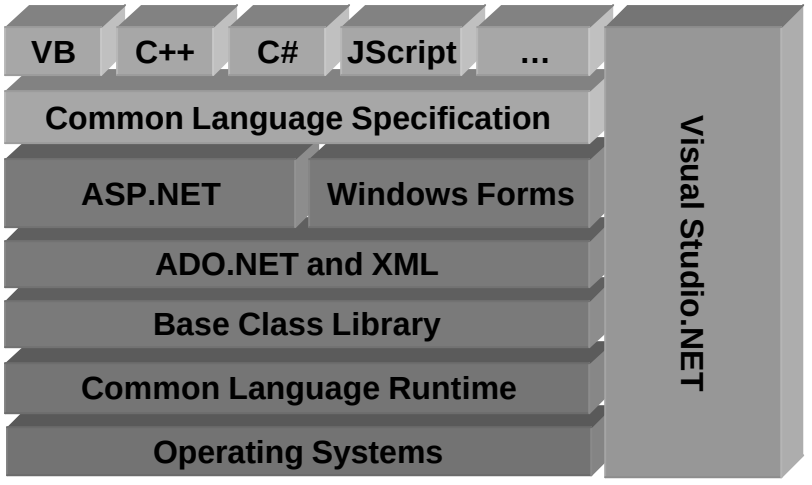


:NET

What is .NET?

- NET is a platform that enables:
  - Software as services, especially over the web
  - Distributed computing
  - Componentization
  - Enterprise services

.NET Platform

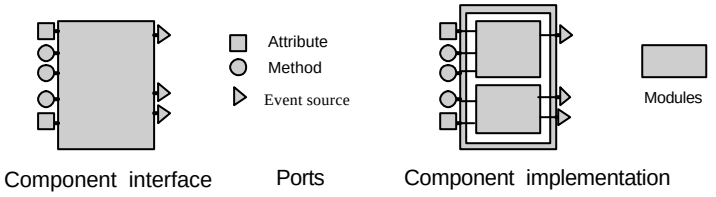


.NET Framework Components

- **Common Language Runtime (CLR)**
  - Common type system for all languages
  - Runtime environment
- **Class libraries (.NET Framework)**
  - Base class libraries, ADO.NET and XML
  - Windows Forms for, Win32 applications
- **Web application platform ASP.NET**
  - Interactive pages
  - Web services that are SOAP enabled

.NET Component Model - Implementation

- **A component (assembly) is made of modules, which are traditional executable files (DLL).**
- **Modules cannot be assemblies, thus the .NET model is not hierarchical.**



Framework

- **.NET relies on the traditional programming approach : the framework is seen as the language run-time support.**
  - MISL – Microsoft Intermediate language (similar to Java Byte code)
  - Common Runtime Language (similar to Java Virtual Machine)
- **Transaction control relies on MTS.**

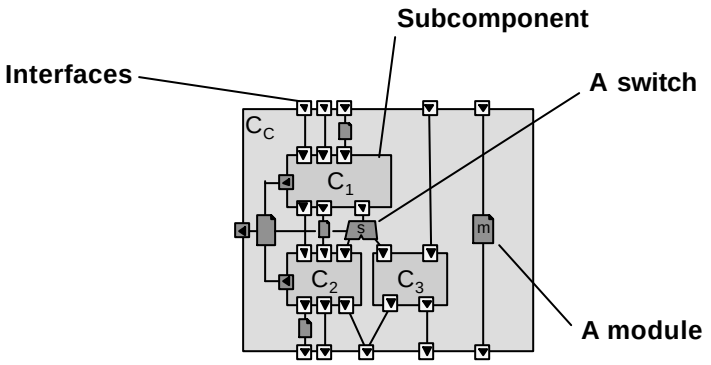
Lifecycle

- **Assemblies (and their modules) are local to an application, and thus different DLLs with same name can run simultaneously.**
- **Each assembly has a versioning information about itself and about the assemblies it depends on.**
  - Version control is delegated to the dynamic loader, which selects the “right” version.
- **Significantly improve the application packaging and deployment.**

Other component models

- OSGI Component Model
- KOALA component model
- IEC 61131-3 standard languages (functional blocks)
- Real-time components

A Koala Component



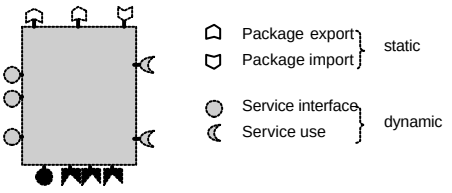
OSGI Component Model

- **Components**
- **Interface of a Bundle Component**
- **Assembly of Bundle Components**
- **Implementation of a Bundle Component**

Components

- **A bundle use three kinds of ports to express its interactions with**
  - Traditional technology
  - Other components
  - The run-time environment
- **Bundles may listen to events published by the framework such as the insertion of a new component in a system.**

Interface of a Bundle Component



Interface of a bundle component

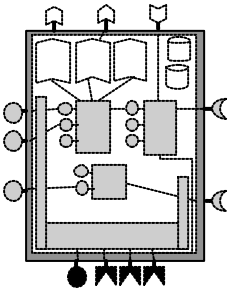
Ports

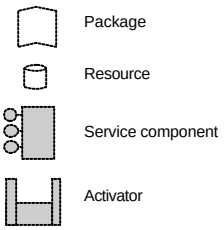
Assembly of Bundle Components

- **A system is an evolving set of bundle components.**
- **A bundle component publishes a service interface**
  - It can attach to it a set of properties describing its characteristics.
- **A component requires an interface for its use,**
  - It will select one via a query expression based on these properties.
- **This flexibility also has its counterpart**
  - There is no guarantee than the service will continue to be available

Implementation of a Bundle Component

- **JAR archive containing:**
  - Service components
  - Java packages
  - Other resources files





Component models