Chapter 4

Component Models and Technology

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

Introduction

- A Short Historical Perspective
- Component Interface and Connections
- Performing Services Transparently

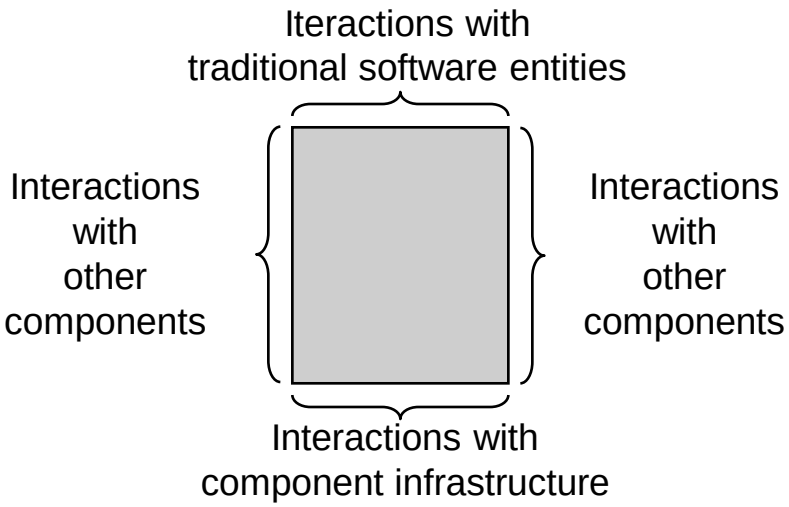
A Short Historical Perspective

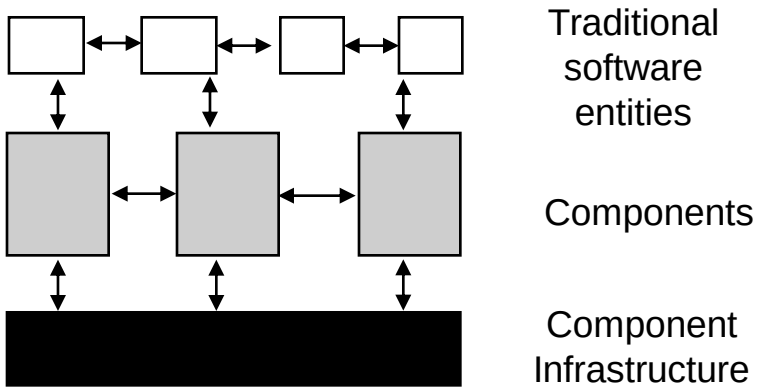
- **Programming languages, can be seen from either**
  - The run-time point of view or,
  - The design and reuse perspective

Component Interface and Connections

- **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.

Component Interactions





Majors steps in CBD lifecycle

Aspect	*Phase*	*Actor*
Interface	**Definition**	Designer
Assembly	Assembly	Architect
Implementation	Implementation	Developer
Lifecycle	Packaging, Deployment	Administrator
Framework, run-time support	Execution	End User

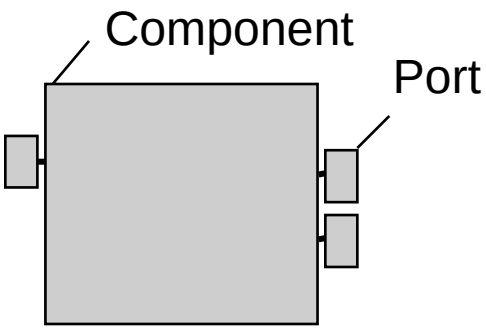
Performing Services Transparently

ACME Architectural Description Language

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

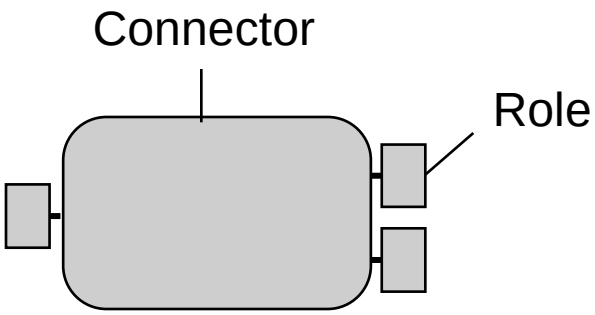
Components and Ports

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



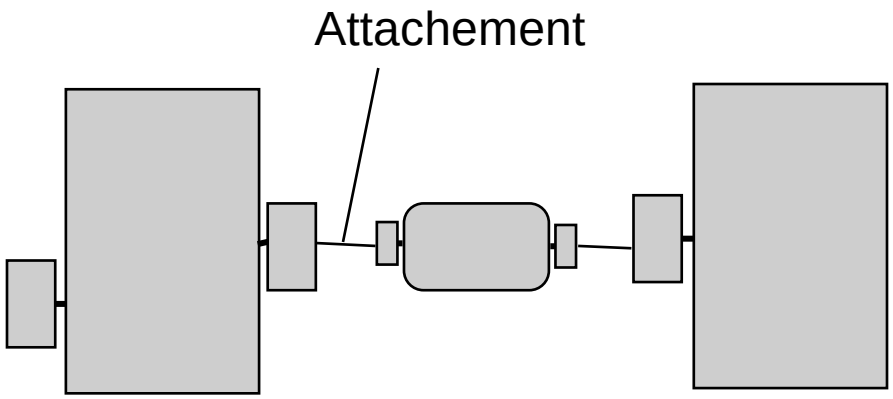
Connectors and Roles

- **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**

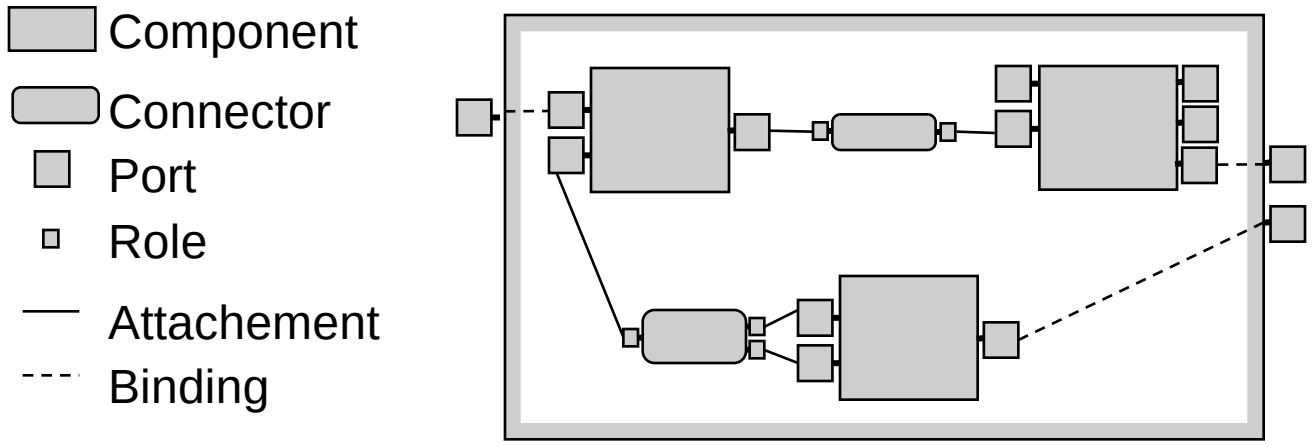


Systems and Attachments

- **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



Java Bean Component Model

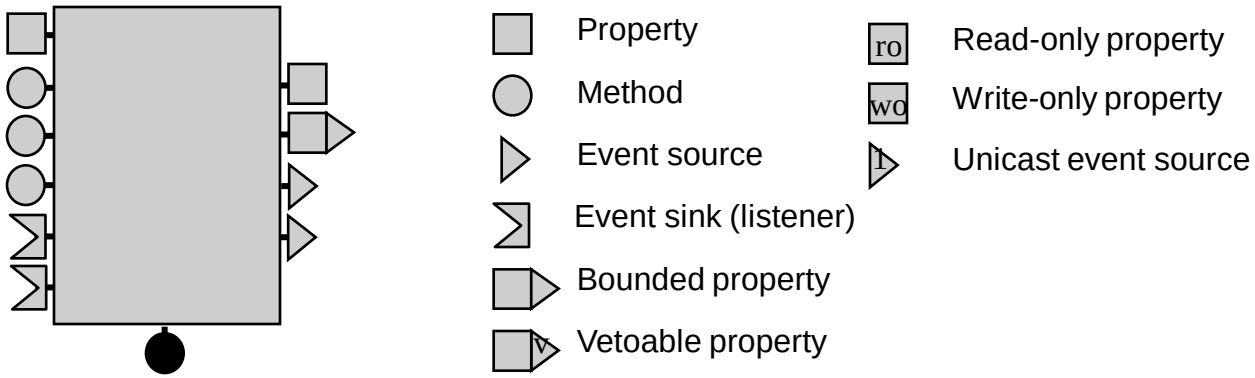
- Key Features
- Interface of a Component
- Implementation of a Component
- Components Assembly
- Packaging and Deployment

Key Features

- **Bean Box**
- **"*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.

Interface of a Component

- **This model defines four types of port:**
  - methods,
  - properties,
  - event sources and
  - event sinks called listeners.

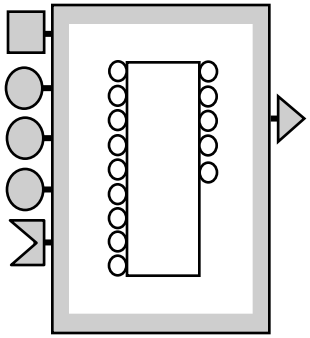


Ports

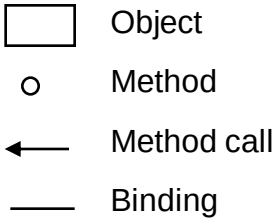
Implementation of a Component

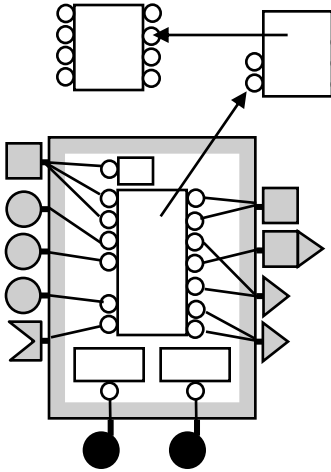
- **Most bean components are implemented by a simple Java object, the object being encapsulated in the component, but there are more sophisticated implementations possible.**
  - Wrapping a legacy object.
  - Multiple-objects implementation.
  - Dependency on traditional entities.

Implementations of Bean Components



A simple implementation

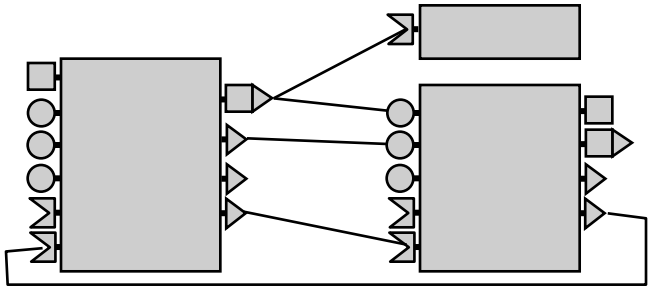




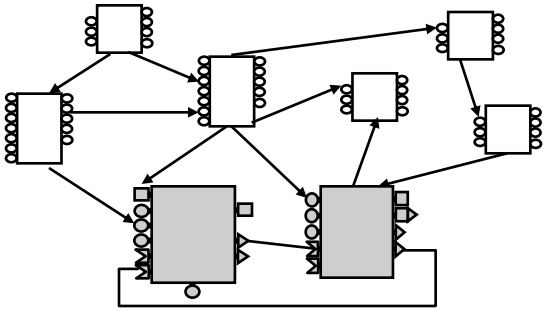
A more complex implementation

Components Assembly

- **Assembly is one of the key features of Java Bean though no not specific solution is provided.**
  - 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.
- **The customization code can be more complex than the component itself!**
- **Each package item can be marked "Design Only", so that they can be removed in a final application.**

Framework : The Container Approach

- Services can be made available to components without having to change that component's source code.