Concepts concerning systems of software components

Prof. Dr. Hans-Gert Gräbe

Institute of Computer Science,
Leipzig University

https://informatik.uni-leipzig.de/~graebe

A First Component Definition

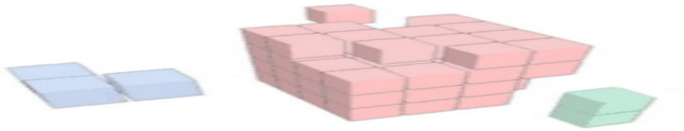
Source: Clemens Szyperski (2002). Component Software

> Software components are executable software units that are independently produced, acquired and configured and from which functioning overall systems can be assembled.

- Composition aspect is in the foreground
  - standards are required for this:
    - Classification in a special **component model**
    - Executable on a special **component platform**
- Systems composed in this way are called **component software**
- Independence and feasibility
  - Independence of development and use
  - Abstractions at source code level are thus excluded
  - Oldest examples of software components: Libraries

Component Properties

Main properties of a component



... a functionally and technically self-contained executable unit

... independent as a unit developable and configurable

A component is...

... reusable

... accessible only via exactly specified interfaces

UNIVERSITÄT LEIPZIG
Institute of Computer Science
Business Information Systems

Commonalities of the success stories

- Existence of an **infrastructure**
  - basic functionality for interoperability is provided to a sufficient extent
- Components have **enough substantial functionality to make** repeated development ineffective
- Components from independent providers can coexist in the infrastructure
  - Composability is more likely than guaranteed
  - plug and play
- Components exist at a level of abstraction that have a **direct meaning for** the distributing client
- This is in contrast to objects, which have no independent meaning for non-programmers
  - but: Object technology is the best way to realise component technology

UNIVERSITÄT LEIPZIG
Institute of Computer Science
Business Information Systems

Different Component Concepts

One has to distinguish between life cycle events:

- Component as a concept
- Component as a deliverable prototypical unit
- Component as a configurable unit in a prototypical system context
- Component as a unit to be distributed, configured and installed in a concrete system context
  - distinguish between "deploy" and "install"
- Component instance as an instance of an installed component

Components and services: The term "service" is often also used in the sense of a connection with a service provider positioned on a market

- Services in this sense are orthogonal to the component concept and can use component instances.
- However, this requires a concrete hardware, software and organisational infrastructure

Summary

***Component technology**:*

Multi-level modular software construction from prefabricated components in which functionality is encapsulated in such a way that it can be reused in a uniform manner.

Requirements:

- Formally sound **component concept** as a basis
- **Description techniques** for such components
- Development of a **process model** for the development, management and composition of components
  - Support for the assignment of different roles
- **Tools** that support the description and the process model
  - for system generation itself
  - to the documentation
  - to verify and secure important and critical system properties

Components and Objects

What are components and what are not?

- "Component Software - beyond object oriented programming"
- Problem of delimiting the terms *object* and *component*
  - Often used synonymously, even as „component object“
- In OO an object is an instance of a class (constructor pattern) or a clone of a prototype object (factory pattern)
- Commonalities:
  - They provide *services* addressed via *interfaces*
  - Interfaces are typed and categorised according to rules
  - Interfaces are described according to pattern (syntax) and meaning (semantics)

Components and Objects

Approach: Fix meanings of terms by specifying characteristic properties

Component definition

Characteristic properties of a component:

1. Unit of independent packaging
2. Unity of composition by third parties
3. Without (externally observable) state

Object definition

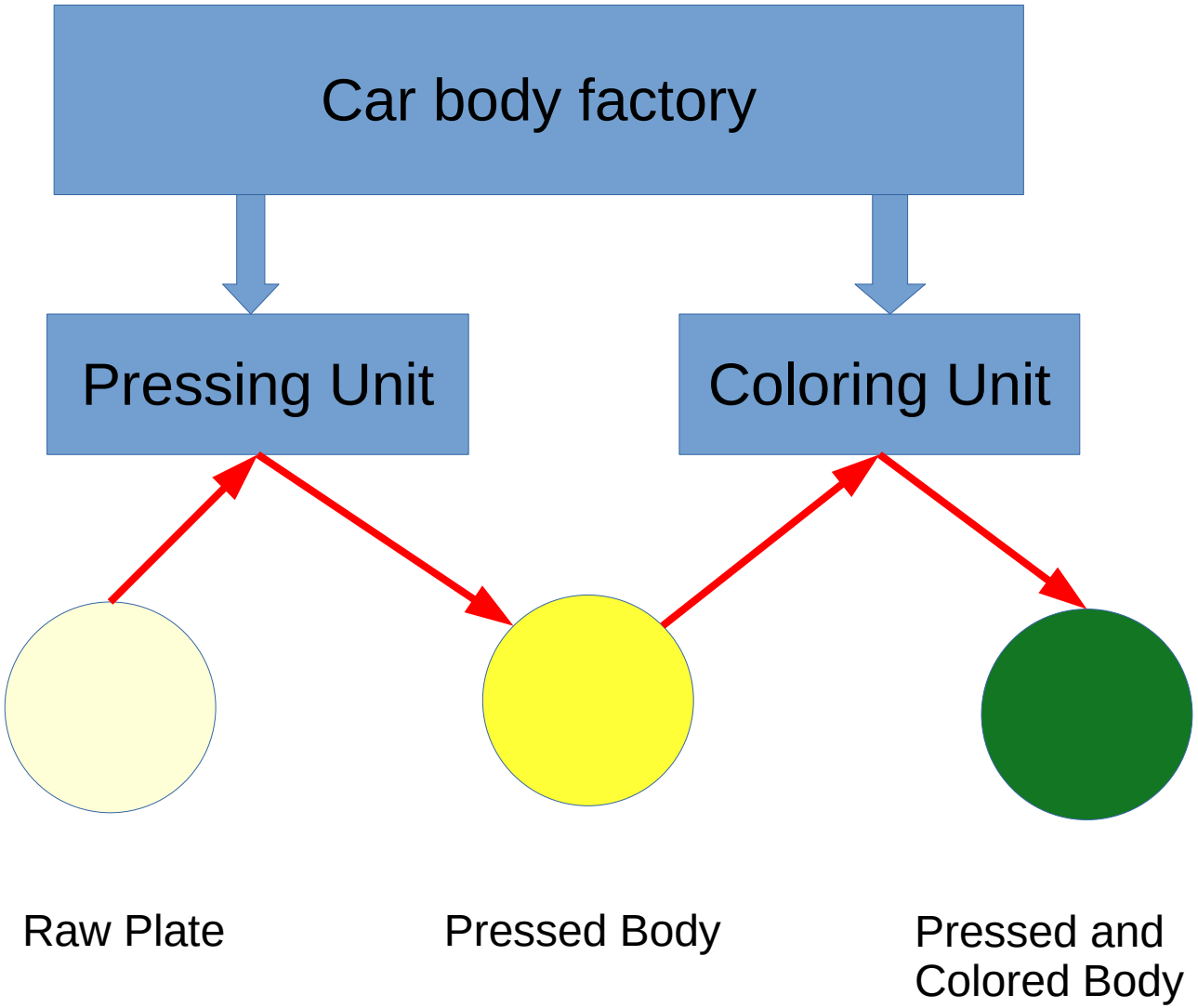
Characteristic properties of an object:

1. Unit of instantiation with (unique) identity
2. Can have (externally observable) state
3. encapsulates state and behaviour

Components and Objects

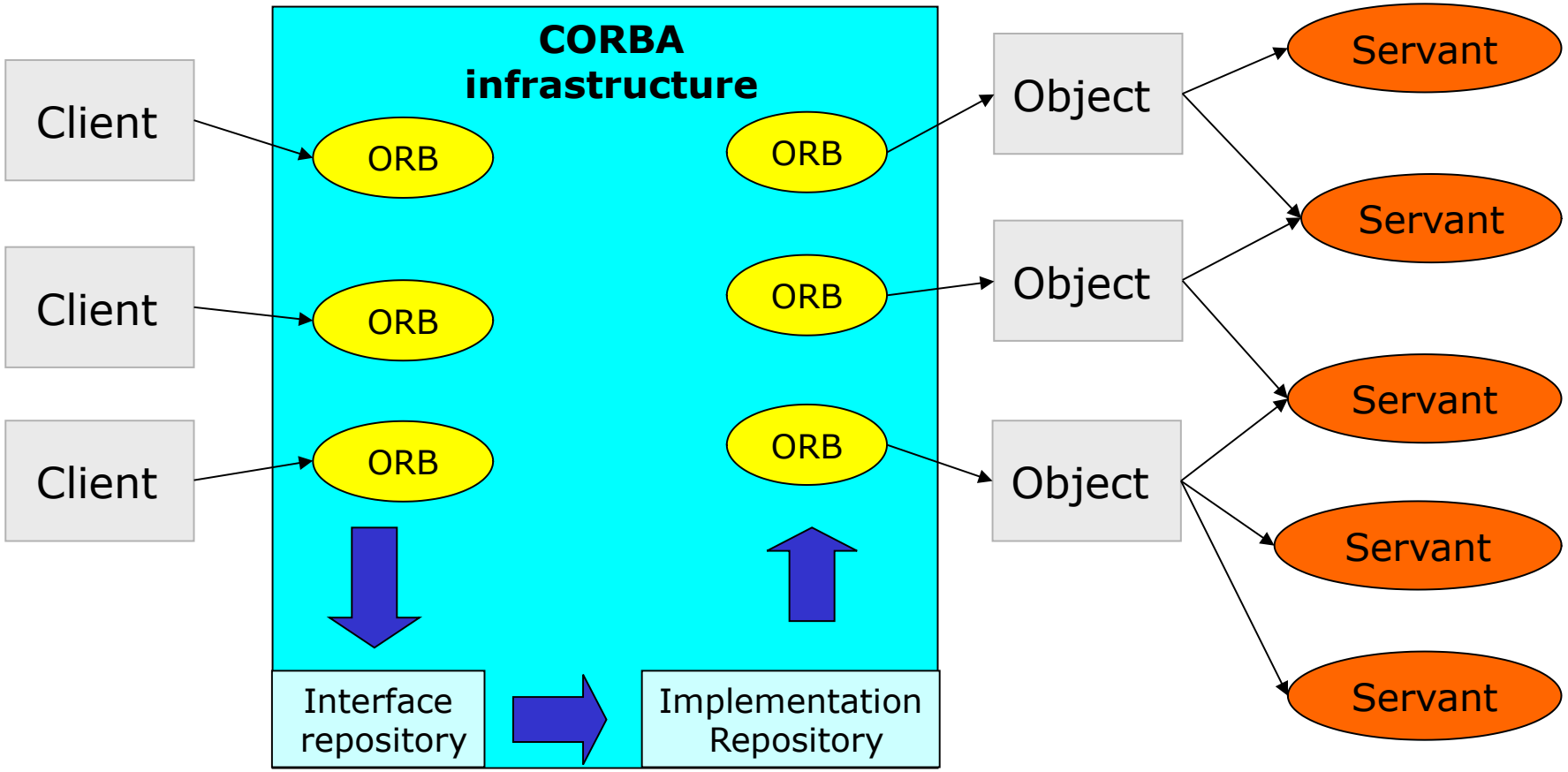
Components

Objects



The Corba Architecture

Architecture at a glance



Corba Services

Delivery of a service requires cooperation of several objects

Example: The Licensing service

- Management of object licences, use mode, billing of usage
- Support for different licensing models
- 2 interfaces: *Licence Service Manager* (LSM) and *Licence Service*
- Object O requests a licensed service, Service creates a Service object SO
- The SO learns from LSM under which conditions its use is legitimised
  - SO requests a reference to a corresponding (manufacturer-specific) licence service object (LSO) from LSM
  - SO informs LSO about **context of** licence request
  - LSO checks which use of the SO is legitimate in the context O
  - LSO initiates transition of SO to permitted state (demo mode, trial mode, if applicable)
  - SO serves O during definite time in definite mode.
  - SO finishes service. SO informs LSO about end of use.
  - LSO records use, creates bill etc.
- Current licence design therefore encapsulated between SO and LSO (user profiles, licence duration and expiry dates etc.)

UNIVERSITÄT LEIPZIG
Institute of Computer Science
Business Information Systems

The CORBA component development model

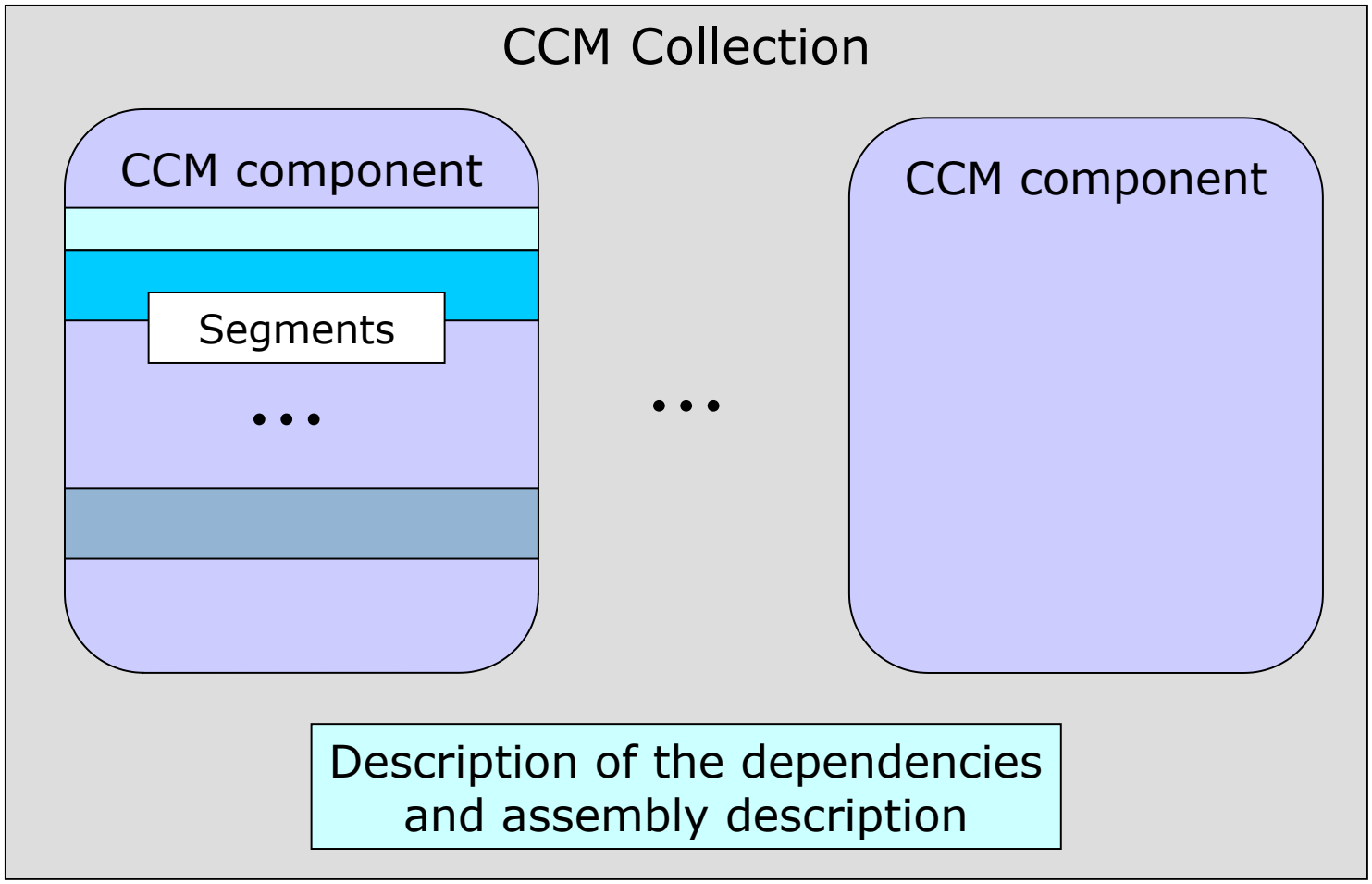
CORBA is standardised by the OMG (Object Management Group), an international industry consortium with more than 800 members.

During 2003-2012 they developed **CORBA 3 with the CCM** as its very center. With that component model, a **development model** with four levels was also specified:

- **Abstract Component Model**: External properties of CCM components and definition of component types in IDL (Interface Definition Language) – **Components as Black Box**
- **Component Implementation Framework**: Programming model for creating component implementations and for describing the relations to implementations of other components – **Components as White Box**
- **Container Programming Model**: Description of the container architecture and the interfaces to the ORB and the components – **Services provided by the Infrastructure**
- **Packaging and Deployment**: Packaging of components and assemblies, specification of descriptors and the deployment process.

The CORBA Component Model

Basic structure of a CCM collection

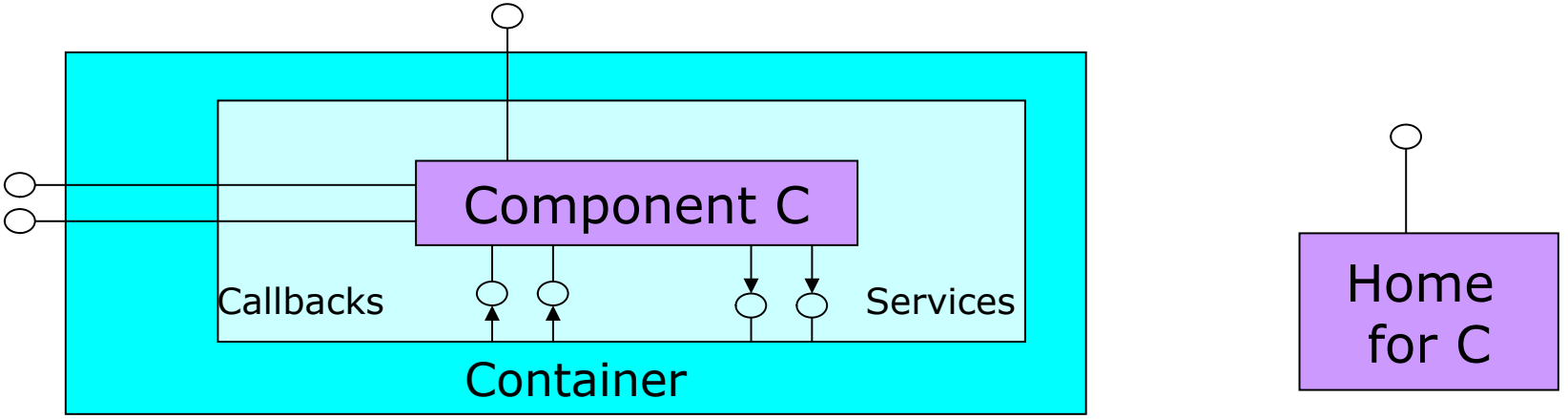


UNIVERSITÄT LEIPZIG
Institute of Computer Science
Business Information Systems

CCM container

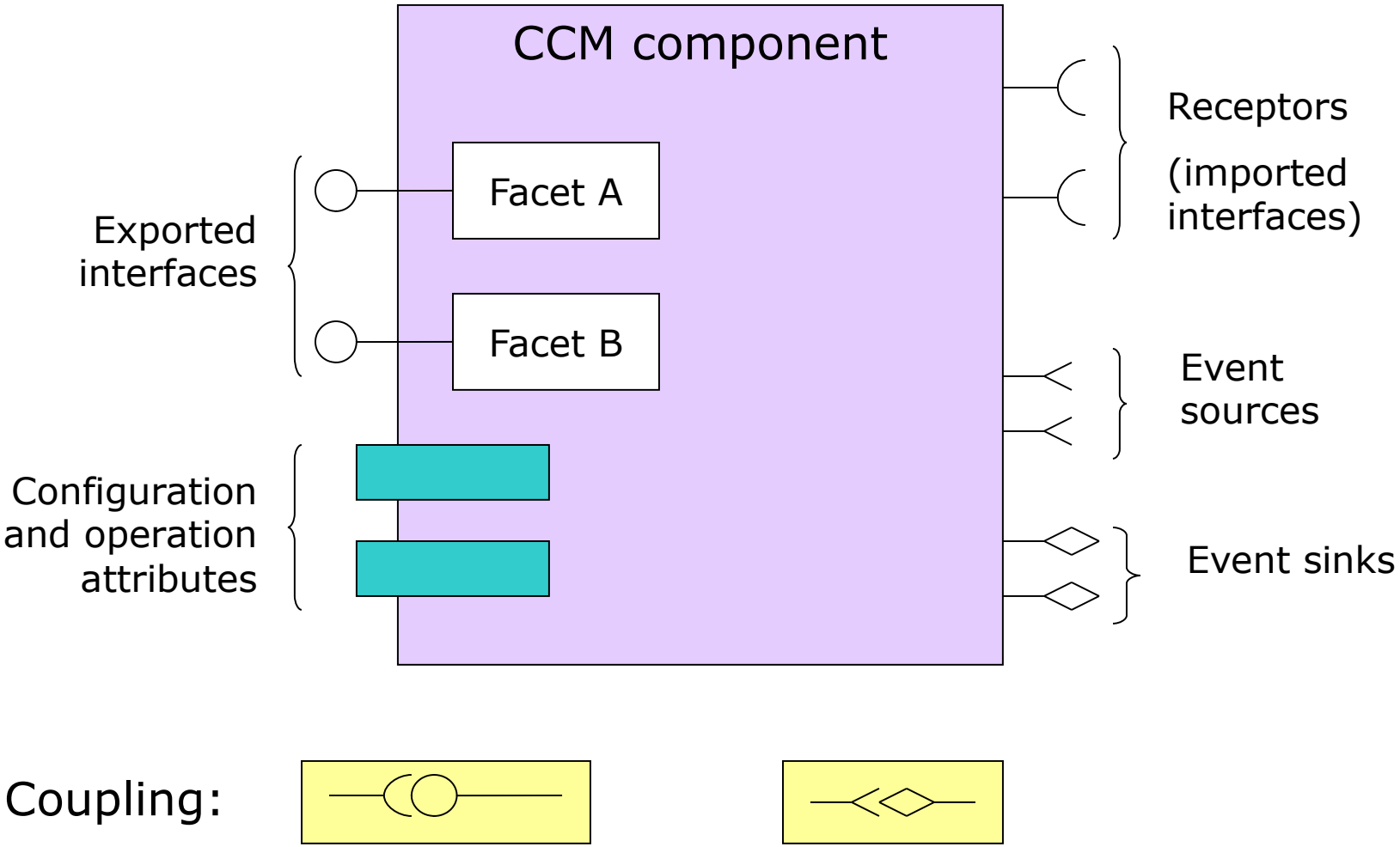
CORBA 3 defines a **component implementation framework** (CIF)

- Generators create code from inputs in **CIDL format** (component implementation description language) that supplements the component code.
- Each component instance is housed in a **CCM container**, via which the facets and receptors are connected. Receptors and services can be bound in such a container via callback.



The CORBA Component Model

Structure of a CCM component



Coupling:

UNIVERSITÄT LEIPZIG
Institute of Computer Science
Business Information Systems

Ports of CCM components

- **Facets** (facets)
  - exported interface, usually assigned to a sub-object of the component
- **Receptors** (receptables)
  - Imported interfaces, internal references to external objects required for component operation
  - connect / disconnect operations
  - can be explicitly required in the assembly description or integrated at runtime
- **Event** sources and event sinks
  - Ports to be connected by event channels
- **Primary key** (entity components only)
- **Configuration** and **operation attributes**
  - Named values that are externally visible via **accessor functions** or **modifiers (**mutator).

The CORBA Component Model

- **Home interface**, via which the component factory can be reached
  - implemented in the component class
  - so component concept different from the one in the lecture
  - Management of the life cycle of component instances
- Special facet **E-interface** (equivalent interface), which can be used to navigate between the facets of the component
  - Clients must support CORBA-3 to take advantage of these navigation options
- Configuration interface
  - Support for the initial configuration of new components
  - special call signal completes the configuration phase
  - Only then are calls to the operational interfaces possible, while calls to the configuration interface are prohibited.

The CORBA Component Model

Pre-packaged **basic services** (pre-packaged object services)

- **Transaction service**: through container or itself
  - Component: Description contains the transaction requirements (supported, required, required new, not supported)
  - Container: Execution of transactions according to the specification of the individual components
- **Persistence service**: by container or self
  - Component: Description of requirements in PSDL format (persistent state description language)
- **Security**:
  - Access rights can be described in CIDL format and checked by the container
  - Notification service: setting up and managing event channels