

# Software components and distributed systems

**INF 5040/9040 autumn 2008**

**lecturer: Frank Eliassen**

Frank Eliassen, SRL & Ifi/UiO

1

## Literature

- G. T. Heineman, W.T. Councill, "Component-based Software Engineering" - Putting the Pieces Together, Addison Wesley 2001, ch 1 and 3
  - copies available at <http://heim.ifi.uio.no/~frank/inf5040/CBSE/>
- TvS, chap 10.1.2 (a bit on EJB)
- Recommended
  - Szyperski, C., Gruntz, D., Murer, S., "Component Software – Beyond Object-Oriented Programming", *Second Edition*, Addison Wesley/ACM Press, 2002

Frank Eliassen, SRL & Ifi/UiO

2

## A history of middleware

- First generation middleware
  - Exclusively based on *client-server model*
  - Examples include Open Group's DCE
- Second generation middleware
  - Based on *distributed object technology*
  - Examples include CORBA and Java RMI
- Third generation middleware?
  - Based on (emerging) *component technology*



Frank Eliassen, SRL & Ifi/UiO

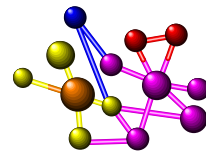
3

## The emergence of component technologies

- What is a component [Szyperski]?

“a unit of *composition* with *contractually specified interfaces* and explicit *context dependencies* only”

“in this context, a component can be *deployed independently* and is subject to *third-party composition*”



Frank Eliassen, SRL & Ifi/UiO

4

## Software component according to Heineman et al

- *Software component*: Software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard
- *Component model*: defines specific interaction and composition standards
- *Component model implementation*: dedicated set of executable software elements required to support the execution of components that conform to the model
- *Software component infrastructure*: a set of interacting software components designed to ensure that a software system constructed using those components and interfaces will satisfy clearly defined performance specifications

Frank Eliassen, SRL & Ifi/UiO

5

## Rationale for components

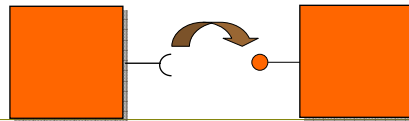
- Time to market
  - Improved productivity/ reduced complexity
  - Focus on reuse
- Programming by assembly rather than by engineering
  - Reduced requirements to knowledge
- Most important advantage: development of server side?
  - (cf. EJB/JEE or CORBA Component Model - later)

Frank Eliassen, SRL & Ifi/UiO

6

## Composition

- Components and composition
  - Composition is the fundamental method for construction, extension and reuse of component-based software development
  - In contrast to (implementation) inheritance in object-oriented approaches



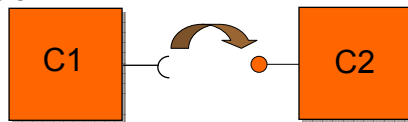
*“Components are made for composition”*

Frank Eliassen, SRL & Ifi/UiO

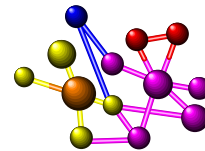
7

## Connection-oriented programming

- Composition of pre-manufactured components
- Binding of incoming and outgoing interfaces
  - provided/required interfaces
  - Reflects direction of method calls
    - Not the direction of data flow
  - Outgoing interface
    - The method calls a component potentially may issue
- Support for distribution?
  - When the binding can be made across address spaces and computers



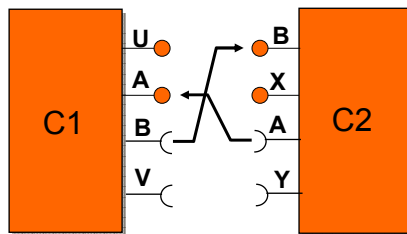
Frank Eliassen, SRL & Ifi/UiO



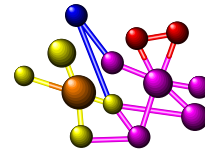
8

## Third party composition

- The composition can be done by a third party external to the components themselves (loading and binding)
- Example
  - Connections (bindings), outgoing and ingoing interfaces
  - Connects (binds) "matching" interfaces
  - Can be done during run time by a third party
    - Can typically be realized by setting an appropriate attribute of the component with the outgoing interface (for C1, methods: setB, setV)

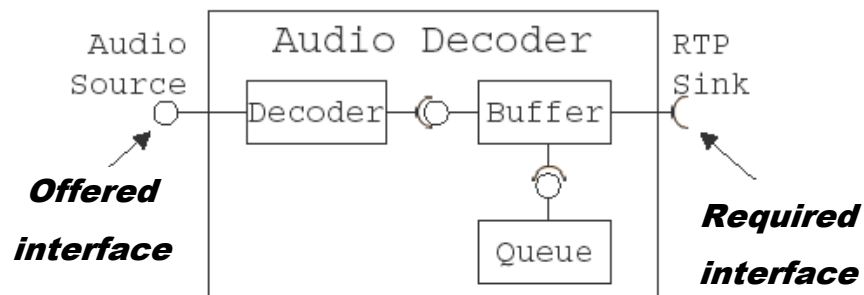


Frank Eliassen, SRL & Ifi/UiO



9

## Composition: Reuse and assembly of components



Frank Eliassen, SRL & Ifi/UiO

10

## Background for Java og CORBA component models

- Known problems with CORBA and Java-RMI
  - How to deploy the components of my application?
  - Which services will be available on a given host?
  - Who activates my objects?
  - Who manages the life-cycle of my objects?

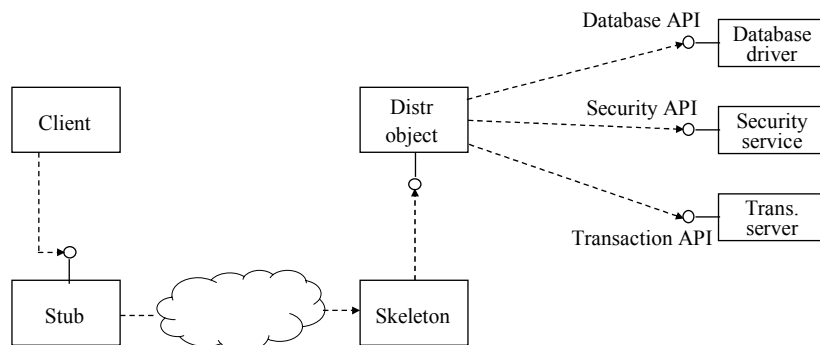
=> *We need a standard development, deployment and runtime environment for distributed objects (CORBA, Java)*

Frank Eliassen, SRL & Ifi/UiO

11

## Explicit middleware: lack of “separation of concerns”

- Programs directly towards a middleware API
- Application logic entangled with logic for life cycle management, transactions, security, persistence, etc.

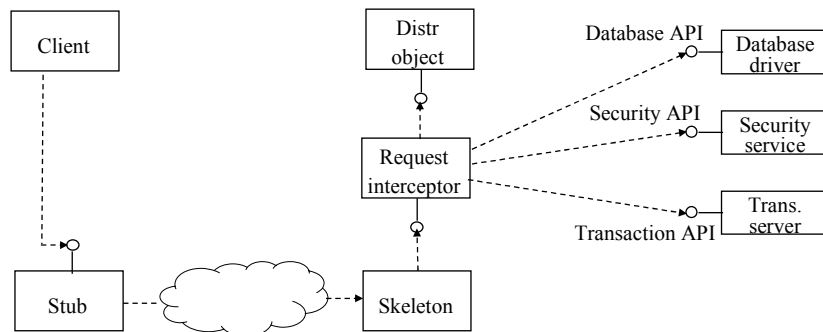


Frank Eliassen, SRL & Ifi/UiO

12

## Implicit middleware: better support for “separation of concerns”

- Logic for life cycle management, transactions, security, persistence, etc. managed by the middleware
- Requirements for middleware services declared separately and can later be changed without changing the application code
- Middleware can be changed without changing the application code



Frank Eliassen, SRL & Ifi/UiO

13

## Component platform

- A standard development, deployment and runtime environment can be designed as a set of contractually specified interfaces
- Contracts agreed between components and a component platform
- Component platform defines the rules for deployment (installation), composition and activation of components.
- For delivering and deploying a component is required a standardized archive format that packages component code and meta-data



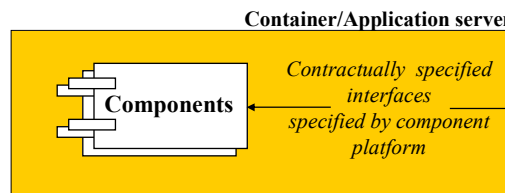
Frank Eliassen, SRL & Ifi/UiO

14

## An implementation of a component platform is often called a *container*

### Responsibilities of the container

- life cycle management
- system services
- security
- dynamic deployment and activation of new components
  - e.g., resolve dependencies dynamically or activate components requested in method calls
  - when a component has a need for a service, the container will load the component that offers the service, dynamically



Frank Eliassen, SRL & Ifi/UiO

15

## Contracts



- What is in a contract?
  - *Set of provided* interfaces.
    - Some of these may be required by the component platforms
  - *Set of required* interfaces.
    - These must be offered by other components available in the container
  - Pre and post conditions/invariants
  - Extra-functional requirements: transactions, security, performance, ...
- Functions defined both syntactically and semantically
  - `int add(int a, int b)`
  - pre:  $a + b \leq \text{Integer.MAXINT}$
  - post:  $\text{result}' = a + b$
- Extra-functional requirements
  - Guarantees: Response within 10 ms
  - Conditions: Needs 1000 CPU-cycles
  - Transaction requirements: e.g, create new transaction when component is invoked, serializable, ...

Frank Eliassen, SRL & Ifi/UiO

16



## Summerizing the elements of a component model

- Interfaces
- Naming
- Meta data (including dependencies)
- Interoperability
- Customization
- Composition
- Evolution support
- Packaging and deployment

Frank Eliassen, SRL & Ifi/UiO

17

## Key players

- OMG and components
  - CORBA v3 standard with CORBA Component Model (CCM)
- Microsoft and components
  - Development of COM/DCOM, COM+ and .NET
- SUN and components
  - Development of Java Beans and EJB



Frank Eliassen, SRL & Ifi/UiO

18

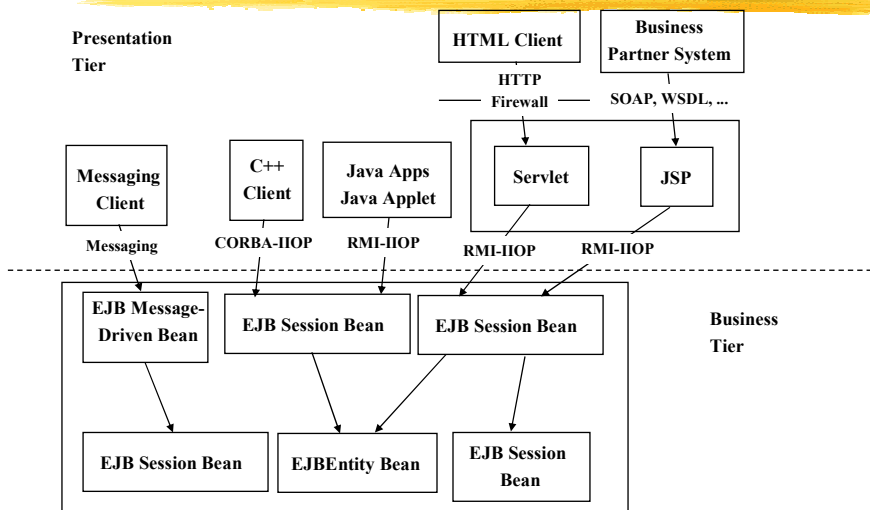
# Enterprise Java Beans (EJB)

- Component architecture for deployable server side components in Java.
- EJB 3.0: based on Metadata facility in Java 5
  - annotations in source code
- Literature: e.g., EJB 3.0 in a nutshell
  - <http://www.javaworld.com/javaworld/jw-08-2004/jw-0809-ejb.html>
- Three types of enterprise beans
  - Session beans (verb)
    - POJO with “session bean” annotations (meta-data)
    - Transient, application logic (business rules ...)
  - Entity beans (noun)
    - POJO with “entity bean” annotation
    - Persistent, data-related logic (updates state of entities)
  - Message driven beans
    - Logic for receiving asynchronous messages and potentially call other beans

Frank Eliassen, SRL & Ifi/UiO

19

# Client-interaction with EJB component system

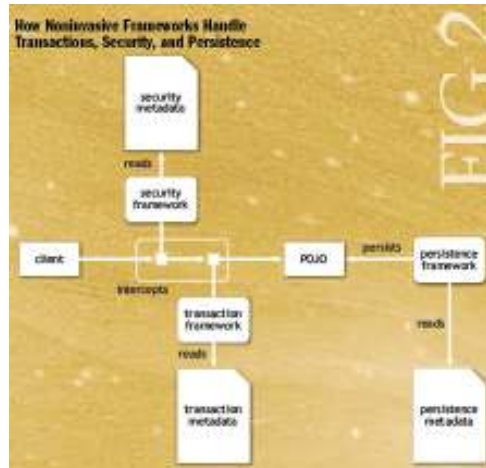


Frank Eliassen, SRL & Ifi/UiO

20

## EJB 3.0 implicit middleware

- Meta data inspected by service framework.
- Necessary "interceptors" weaved in
- Use of "interceptors" to perform system level functions at runtime
- Persistence specified by annotating the relevant attributes in the source code and mapping to database (O/R mapping)



Frank Eliassen, SRL & Ifi/UiO

21

## Connection-oriented programming and EJB

- No support for connection-oriented programming!!
  - Follows traditional object-oriented composition (third party can not bind EJBs, but an EJB can specify dependencies to other components)
  - A strength is automatic composition of component-instances with appropriate services and resources that component-instances are dependent on
    - Automatic configuration of necessary implicit middleware services based on needs specified by annotations or in the deployment-descriptor (transactions, persistence and security)
  - (JavaBeans do have support for connection-oriented programming)



Frank Eliassen, SRL & Ifi/UiO

22

# CORBA Component Model (CCM)

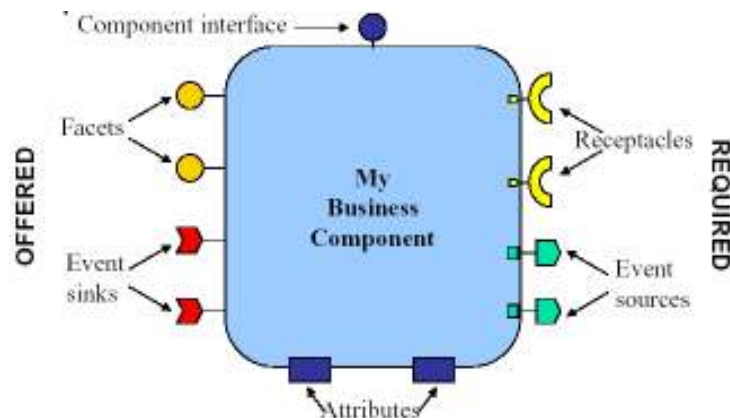
- What is CCM?
  - A *language independent* component model for the server side of a multi-tiered architecture that supports implementation, management, configuration and deployment of CORBA applications
- Important properties
  - An underlying *component model*
  - A packaging technology for deployment of binary, *multi-lingual* executable units
  - A *container framework* that offers implicit middleware for security, transactions, persistence and event based communication

Frank Eliassen, SRL & Ifi/UiO

23

## A CORBA component

- Support for connection-oriented programming
  - Connect/disconnect operations on Receptacles
  - Or based on scripting-language (part of CCM deployment descriptor)

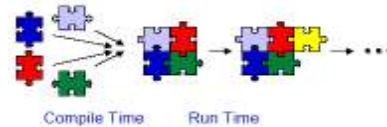


Frank Eliassen, SRL & Ifi/UiO

24

## Composing adaptive software using components

- Importance and interest in adaptive software is increasing dramatically
  - mobile, ubiquitous and autonomic computing
- Components play a major part
- Compositional adaptation
  - dynamic adaptation of architecture of component-based application
    - change component impl
    - redeploy component
    - parameter adaptation
    - change overall architectural framework
    - combinations of the above
- More later (student presentations)



Frank Eliassen, SRL & Ifi/UiO

25

## Summary

- Components
  - Programming according to LEGO-principle
  - Contractually specified interfaces and composition
  - Support for connection oriented programming
- Component architecture
  - Contractually specified interfaces between components and application servers
  - Realizes "implicit middleware"
  - Java: EJB, CORBA: CCM, Microsoft: COM+/.NET

Frank Eliassen, SRL & Ifi/UiO

26