Distributed Systems

Application Architectures

Prof. Dr. Oliver Hahm

Frankfurt University of Applied Sciences
Faculty 2: Computer Science and Engineering
oliver.hahm@fb2.fra-uas.de
https://teaching.dahahm.de

22.06.2023

Development of Commercial IT Solutions

What are the main drivers/-goals when developing IT solutions in a commercial context?

Main Driver of Commercial IT Products

- High flexibility ($\rightarrow$ Ability to adapt)
  - Flexible modelling of today's and prospective business processes
  - Reduction of development time (time-to-market)
  - Integration of existing (partial) solutions
  - Interoperability with third-party components
  - Considering current technological trends:
    - Internet of Things
    - Cloud Computing
    - Big Data
- Low costs
  - Reduction of development costs
  - Reduction of operation, maintenance, and management costs

    $\rightarrow$ Total cost of ownership

Development of Commercial IT Solutions

How can we get there?

Approaches

- Open systems (vendor independence)
- Standard solutions (instead of proprietary development)
- Client/Server models and distributed computing
- Middleware
- Web services
- Application server
- Software reuse and componentware
- Reuse of services/Service Oriented Architectures (SOA)

Which standards/protocol may help here?

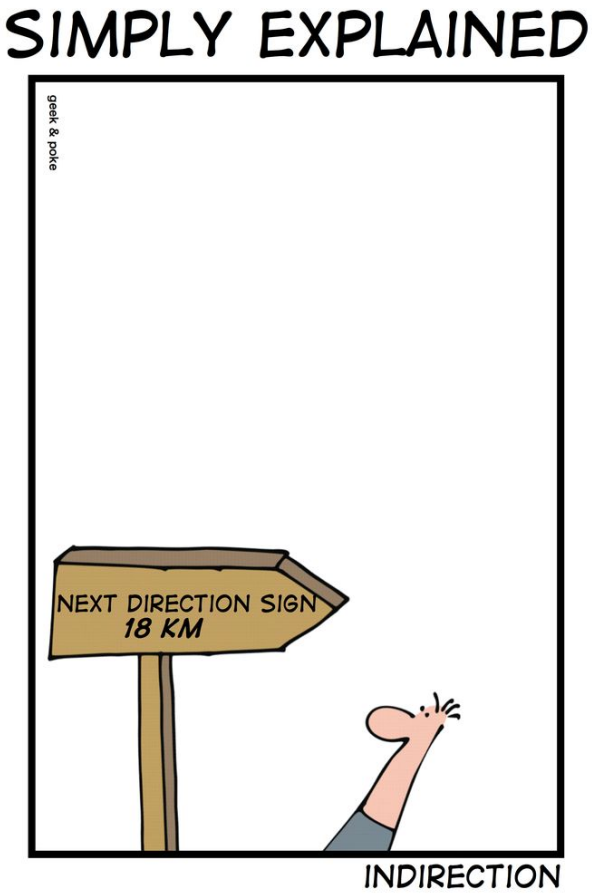
Agenda

- Middleware based Architectures
  - Message orientation
  - Service Orientation
  - Object Orientation
  - Component Orientation
  - Service Oriented Architecture
- Basic Architecture Models
  - Client/Server Model
  - P2P-Modell
  - Multi-Tier Model

Fundamental theorem of software engineering

What is a general approach in computer science to tackle complex problems?

Fundamental theorem of software engineering

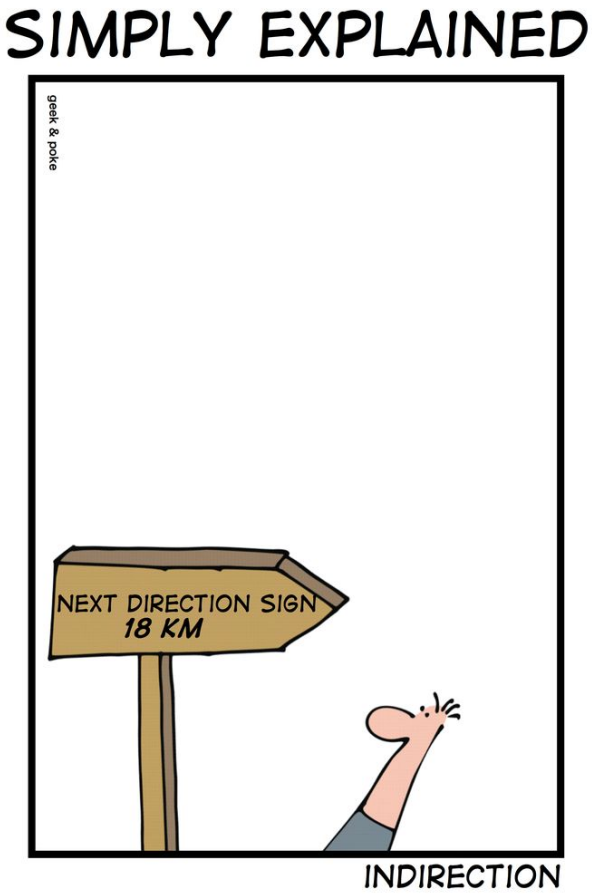


Source: Geek & Poke, Oliver Widder

Theorem

"We can solve any problem by introducing an extra level of indirection."
David J. Wheeler

Fundamental theorem of software engineering



Source: Geek & Poke, Oliver Widder

Theorem

"We can solve any problem by introducing an extra level of indirection."
David J. Wheeler

...except for the problem of too many levels of indirection.

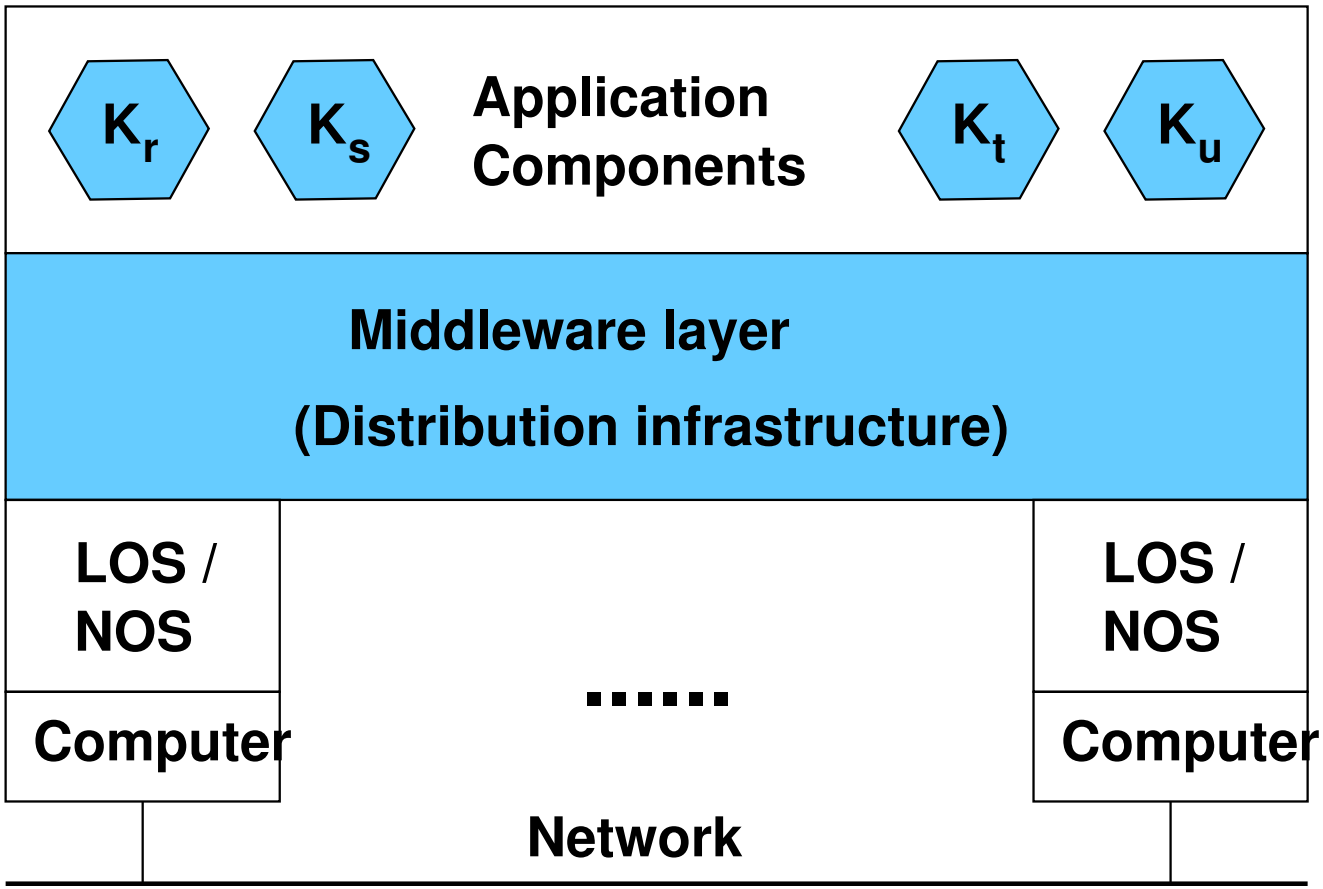
Agenda

- Middleware based Architectures
  - Message orientation
  - Service Orientation
  - Object Orientation
  - Component Orientation
  - Service Oriented Architecture
- Basic Architecture Models
  - Client/Server Model
  - P2P-Modell
  - Multi-Tier Model

What is the role of Middleware?

Tasks of the Middleware

- Software layer as distribution platform for the integration of program components



Middleware Architectures

Each middleware can be characterized by a certain architecture paradigm along with its structural and activity model

Middleware Architectures

Each middleware can be characterized by a certain architecture paradigm along with its structural and activity model

- **Structural model** defines ...
  - the distributable units (program components)
  - their naming and addressing
  - potential auxiliary components

Middleware Architectures

Each middleware can be characterized by a certain architecture paradigm along with its structural and activity model

- **Structural model** defines ...
  - the distributable units (program components)
  - their naming and addressing
  - potential auxiliary components
- **Activity model defines the dynamics** and as such the ...
  - the stakeholders
  - interaction pattern
  - communicated units
  - synchronization
- Implementing a middleware requires access to the components of the underlying layers (esp. the OS)

Middleware Properties

- Degree of *specialization* can be differ a lot, e.g., ...
  - support of a generic cooperation approach
    ($\rightarrow$ main focus of our course)
  - database centric
    (SQL middleware, transaction processing monitor)
  - document or workflow oriented

Middleware Properties

- Degree of *specialization* can be differ a lot, e.g., ...
  - support of a generic cooperation approach
    ($\rightarrow$ main focus of our course)
  - database centric
    (SQL middleware, transaction processing monitor)
  - document or workflow oriented
- Dependence on programming languages
  - Sometimes very high (e.g., only usable with Java)

Middleware Properties

- Degree of *specialization* can be differ a lot, e.g., ...
  - support of a generic cooperation approach
    ($\rightarrow$ main focus of our course)
  - database centric
    (SQL middleware, transaction processing monitor)
  - document or workflow oriented
- Dependence on programming languages
  - Sometimes very high (e.g., only usable with Java)
- Dependence on the underlying OS
  - Often less strong

Middleware Properties

- Degree of *specialization* can be differ a lot, e.g., ...
  - support of a generic cooperation approach
    ($\rightarrow$ main focus of our course)
  - database centric
    (SQL middleware, transaction processing monitor)
  - document or workflow oriented
- Dependence on programming languages
  - Sometimes very high (e.g., only usable with Java)
- Dependence on the underlying OS
  - Often less strong
- Dependence on the underlying hardware
  - Typically very low

Evolution

- Message orientation
- Service orientation
- Object orientation
- Component orientation
- Service Oriented Architecture (document orientation)

→ Surveyed in the following

Agenda

- Middleware based Architectures
  - Message orientation
  - Service Orientation
  - Object Orientation
  - Component Orientation
  - Service Oriented Architecture
- Basic Architecture Models
  - Client/Server Model
  - P2P-Modell
  - Multi-Tier Model

Paradigm: Message orientation

- Basic model of communicating processes ($\rightarrow$ IPC) of traditional OS adapted to a distributed system environment
  - **Processes** as distributable units
  - **Messages** as communicated units
- **Message-oriented Middleware (MOM)**
  - Typically support for persistence and transactions
  - Examples:
    - IBM Websphere MQ
    - Java Messaging Service (JMS) (Teil von J2EE)
    - RabbitMQ

What could you use to realize a MOM?

Example: Socket Programming

- Berkeley Sockets (UNIX)
- Winsock (MS Windows sockets API)
  - Library that basically adopts the UNIX/BSD functions
- Sockets are today the de-facto standard, sometimes via decorated by libraries or classes
- Java Sockets (java.net)
  correspond mostly the model of Berkeley sockets

Agenda

- Middleware based Architectures
  - Message orientation
  - Service Orientation
  - Object Orientation
  - Component Orientation
  - Service Oriented Architecture
- Basic Architecture Models
  - Client/Server Model
  - P2P-Modell
  - Multi-Tier Model

Services

What is a Service?

Services

What is a Service?
How can you provide services in a distributed system?

Paradigm: Service Orientation

Foundation: Remote Procedure Call (RPC)

- Services as **distributable units**
- **Service**: set of provided operations/functions
- Use of remote services via procedure calls
- Typically synchronous processing
- Communicated units are **requests and responses** (containing typed parameters etc. using a common network representation)
- Foundation for client-server applications
- Binding of client and server rather static

Common RPC Platforms

SunRPC

- public domain, available for many systems
- Importance is decreasing
- But the still widely used network file system (NFS) is based on SunRPC

Common RPC Platforms

SunRPC

- public domain, available for many systems
- Importance is decreasing
- But the still widely used network file system (NFS) is based on SunRPC

OSF DCE RPC, Microsoft RPC

- DCE (Distributed Computing Environment): first feature rich service environment
- Too complex for use
- Microsoft RPC mostly compatible with DCE RPC
- Today hardly used any more

Common RPC Platforms

SunRPC

- public domain, available for many systems
- Importance is decreasing
- But the still widely used network file system (NFS) is based on SunRPC

OSF DCE RPC, Microsoft RPC

- DCE (Distributed Computing Environment): first feature rich service environment
- Too complex for use
- Microsoft RPC mostly compatible with DCE RPC
- Today hardly used any more

Apache Thrift

- Very flexible RPC system
- Support for all relevant programming languages
- Widely used

Agenda

- Middleware based Architectures
  - Message orientation
  - Service Orientation
  - Object Orientation
  - Component Orientation
  - Service Oriented Architecture
- Basic Architecture Models
  - Client/Server Model
  - P2P-Modell
  - Multi-Tier Model

Paradigm: Object Orientation

- Objects (in the meaning of OOP) as **distributable units**
- Application := distributed object network
- Interaction by method invocation (with location and access transparency), based on a RPC mechanism
- Reuse of classes on the source code level
- Most relevant platforms
  - OMG CORBA
  - Microsoft DCOM
  - Java RMI

Example: RMI

- **Java Remote Method Invocation (RMI)** (Sun/Oracle)
  - Rather young platform
  - Simple use
  - Supports only the homogeneous world of distributed Java objects

Example: Microsoft DCOM

- Microsoft **DCOM**
  - Extension of COM/OLE via Microsoft RPC
  - Mostly proprietary platform
  - Handed over to Open Group in 1999
  - Subsequently Microsoft services were based on .NET
  - Decreasing importance, but still used in automation

Example: OMG CORBA

- **Object Management Group (OMG)**
  - international non-profit organisation of manufacturers, software components, and users
  - Founded in 1989
    - 3Com, American Airlines, Canon, Data General, HP, Philips, Sun, Unisys, ...
  - 1.000+ members (companies, organizations, universities . . .)
  - Rather fast moving standard body
  - Open, formal standardization process based on Request for Proposals (RFPs)
  - Goal: definition of interfaces, not product development
  - `http://www.omg.org`: freely available documents
  - Still relevant wrt. . . .
    - UML standardization
    - Model Driven Architecture (MDA)

CORBA (Common Object Request Broker Architecture)

- *Independent* from architecture, OS, or programming language
- CORBA IDL is the *interface description language* (resembling C++ syntax)
- Interoperable Object Reference (IOR) as system wide *object reference*
- General/Internet Inter-ORB Protocol (GIOP/IIOP) as *message protocol*
- Many object oriented services and implementations available
- Hardly used for new business applications, but still maintained

Agenda

Paradigm: Component Orientation

- Components as **distributable units**
- Strong independence and interchangeability of the components
- Interaction via **method calls** (based on RPC)

Paradigm: Component Orientation

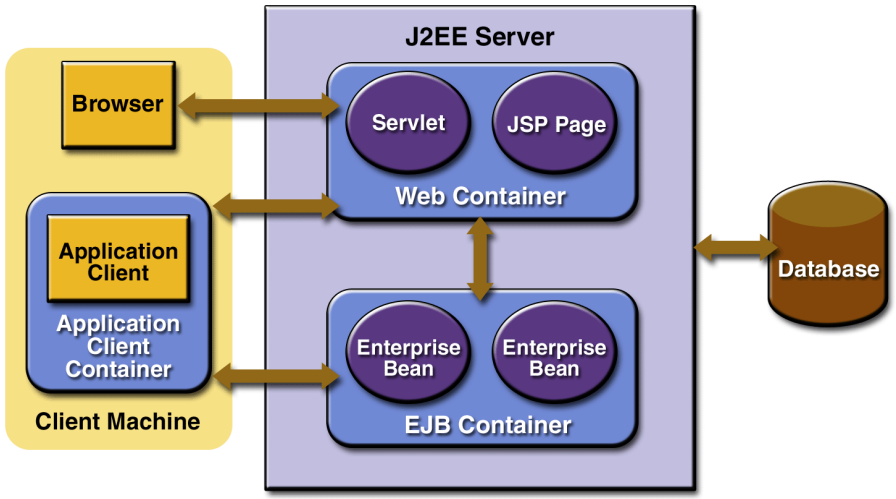
- Components as **distributable units**
- Strong independence and interchangeability of the components
- Interaction via **method calls** (based on RPC)
- Overcoming limitations of object oriented middleware:
  - Implicit dependencies
  - Leaking low-level details $\rightarrow$ lack of transparency
  - Missing support for deployment
- Interfaces are specified in terms of contracts
  - *Provided* interfaces
  - *Required* interfaces
- Rather heavyweight

Jakarta Enterprise Beans

Jakarta Enterprise Beans (EJB) (formerly **Enterprise JavaBeans**) is the most commonly used component model along with Microsoft .NET

- Part of the specification of Java interfaces for server-side components (J2EE/JEE)
- Tightly coupled with **CORBA**
- **Goal:** Simplified application development
- **Application server** as integrated infrastructure for transaction oriented business applications
- Interfaces to standardized services (persistence, transaction management, directory serivces, messaging), bound at **deployment time**
- High scalability for **server side web applications**

Enterprise Java Beans - Component Model



http://www.rizzimichele.it/enterprise-java-beans-and-all-j2ee/

- Components
  - Stateless and stateful **session beans** (Execution of a task for a client without resp. with memory for this client)
  - **Entity Beans** (Representation of business objects in persistent memory, support for transactions)
  - **Message-driven Beans** (asynch. processing of messages, JMS-API)

Agenda

- Middleware based Architectures
  - Message orientation
  - Service Orientation
  - Object Orientation
  - Component Orientation
  - Service Oriented Architecture
- Basic Architecture Models
  - Client/Server Model
  - P2P-Modell
  - Multi-Tier Model

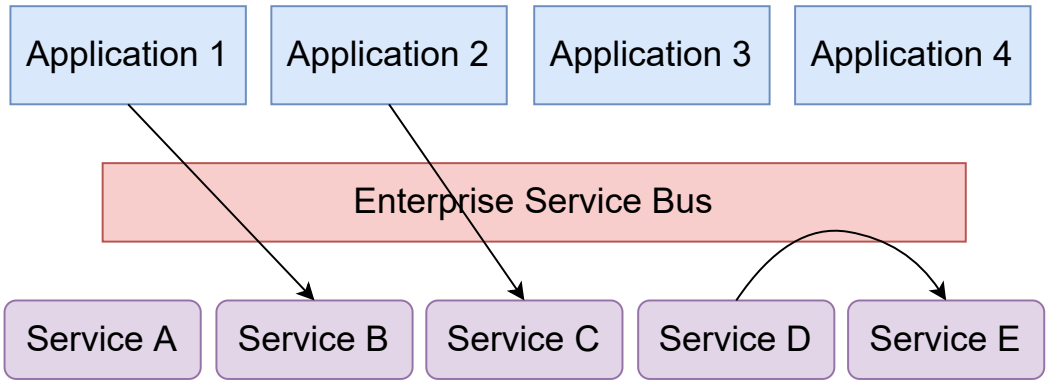
Paradigm: Service Oriented Architecture (SOA)

- Architectural approach for **business applications**
- For structuring and the use of distributed services under potentially differing governance
- **Goal:** achieve technical structuring of application sets
- **Expected benefits:**
  - Definition of services by the means of the business process
  - At the same time multiple use of services in different applications

  $\Rightarrow$ Maintenance reduction
  - Central integration of various applications instead of pairwise interfaces

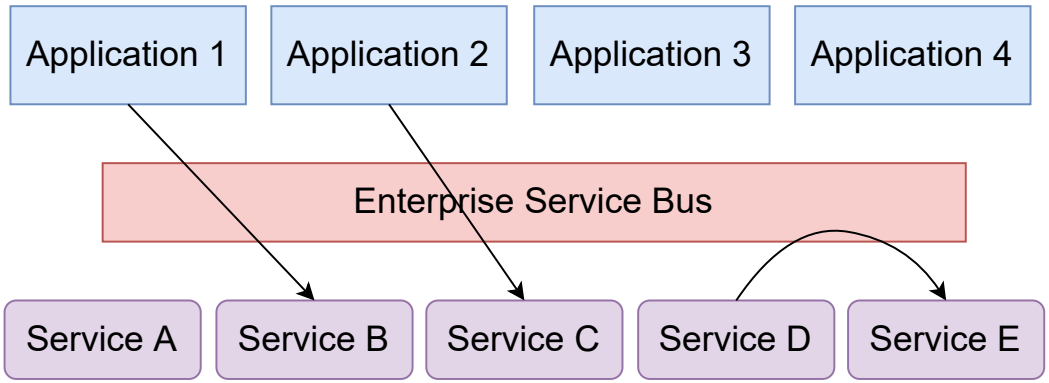
SOA: Services - Applications

- *Ideal image:*



SOA: Services - Applications

- *Ideal image:*



- *Challenges:*
  - Complete decomposition of existing applications is difficult, costly, and not visible for the user
  - Changes to central services affect many applications
  - Formalizing business processes via services is difficult for departments

SOA: Technical View

- Autonomous services described with formal interfaces (**service contracts**) in XML schema documents
- Services do not hold any state whenever possible
- XML documents as communicated units (**messages**)
- Service descriptions (meta data) in a directory (**service registry**)
- Services can be identified and accessed via their descriptions dynamically ($\rightarrow$ no linking required)
- Programming language or technology is irrelevant
- SOA services are currently often implemented as web services
- **Enterprise Service Bus (ESB)** for loose coupling of services

WSDL (Web Service Description Language)

- **Interface/contract description language:**
  - Types
  - Messages
  - Interfaces
  - Services
- W3C standard
- XML based

SOAP

- SOAP (formerly Simple Object Access Protocol)
  - W3C standard
  - **XML document** based interaction framework for **web services**
    - **SOAP Messages** (Envelopes with opt. header and body)
    - Asynchronous processing possible
    - SOAP request/response messages for RPC style
  - Protocol binding framework allows for various underlying transport services, besides HTTP(s), e.g., also SMTP or JMS
  - Java API for XML Web Services (JAX-WS) is part of Java SE

Agenda

- Middleware based Architectures
  - Message orientation
  - Service Orientation
  - Object Orientation
  - Component Orientation
  - Service Oriented Architecture
- Basic Architecture Models
  - Client/Server Model
  - P2P-Modell
  - Multi-Tier Model

Example: Web shop

Which stakeholders are involved?

Which use cases must be realized?

What are the functional requirements?

Basic Architecture Models

Basic architecture models for complex distributed applications

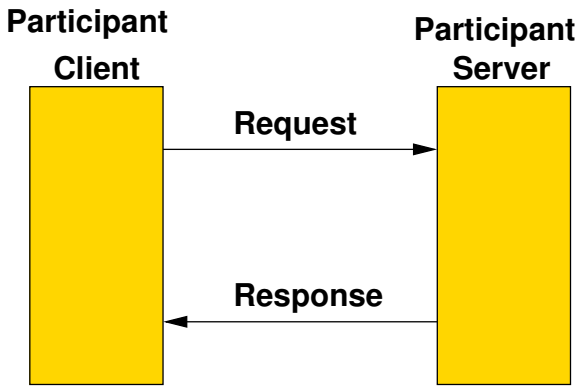
1. Client/Server model
2. Peer-to-peer (P2P) model
3. Multi-Tier model
4. SOA model

Agenda

- Middleware based Architectures
  - Message orientation
  - Service Orientation
  - Object Orientation
  - Component Orientation
  - Service Oriented Architecture
- Basic Architecture Models
  - Client/Server Model
  - P2P-Modell
  - Multi-Tier Model

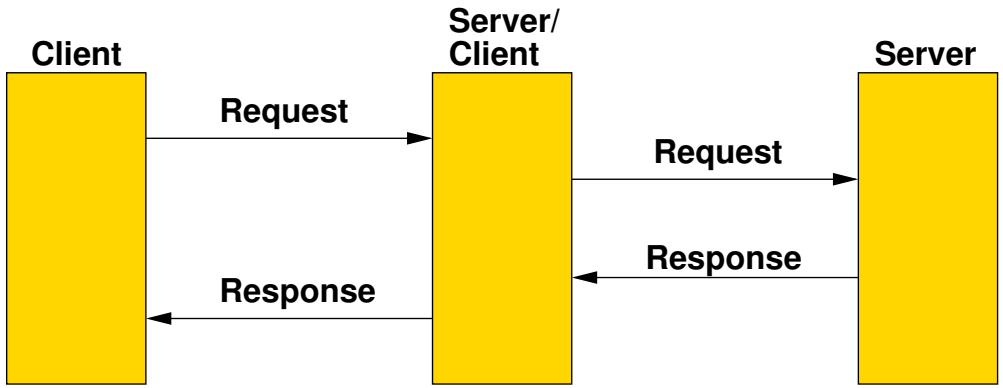
Client/Server Model (1)

- Two different roles
  - Server: Service provider, e.g., web server delivers web pages
  - Client: Service user, customer, consumer, e.g., web browser requesting web pages
- Client and server run typically on different computers



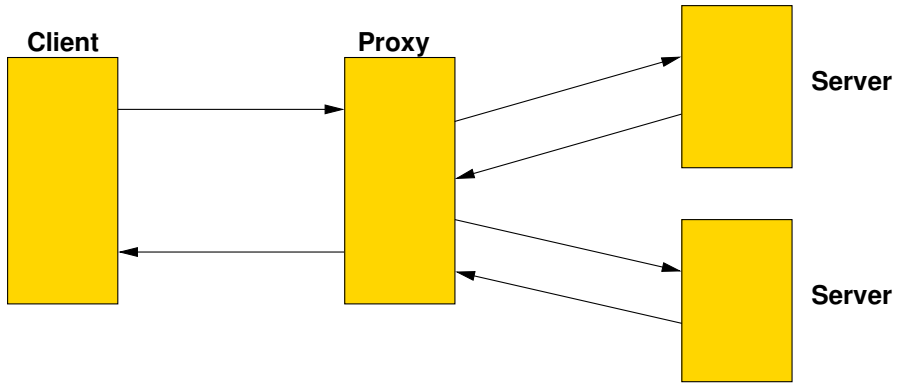
Client/Server Model (2)

- Communication processes are based on request/response interaction pattern
- Initiated by the client
- A client can interact with multiple servers over time
- A server may process requests for multiple clients
- A server may act as a client towards other servers (change of role):



Proxy

- **Intermediary instance**
- Acts as a server towards the client
- Acts as a client towards the actual servers
- Tasks are, e.g., caching, modification of requests ...
- Example: proxy server for web pages



Agenda

- Middleware based Architectures
  - Message orientation
  - Service Orientation
  - Object Orientation
  - Component Orientation
  - Service Oriented Architecture
- Basic Architecture Models
  - Client/Server Model
  - P2P-Modell
  - Multi-Tier Model

Peer-to-Peer Model (P2P)

- **Decentralized** communication between peers
- No additional infrastructure (e.g., servers) required
- Basis for **ad-hoc communication**
- Can be implemented at the network or application level
- Arbitrary message oriented interaction
- Examples
  - File-Sharing, e.g., BitTorrent, Gnutella, eMule
  - P2P development platforms JXTA, MSP2P

Agenda

- Middleware based Architectures
  - Message orientation
  - Service Orientation
  - Object Orientation
  - Component Orientation
  - Service Oriented Architecture
- Basic Architecture Models
  - Client/Server Model
  - P2P-Modell
  - Multi-Tier Model

Example: Web shop

Which Basic functionalities are required?

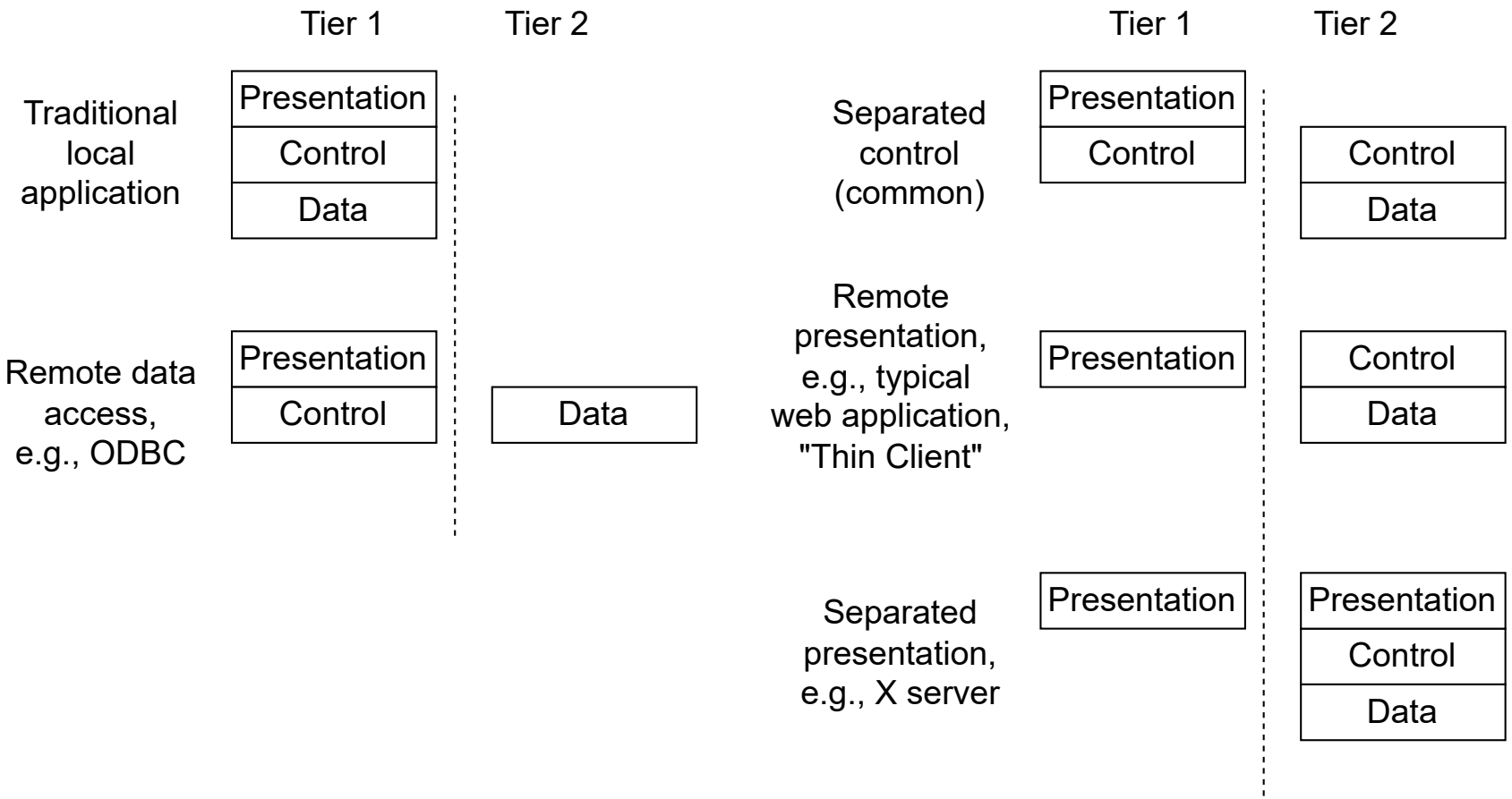
How would you realize these functionalities in a distriButed system?

Multi-Tier Model

- **Tiers** are rather orthogonal wrt (abstraction) layers, typically oriented to
  - User interface/presentation
  - Application control/logic/function
  - Data storage
- No predetermination to used middleware
- Very common today
  - Two-Tier architecture
  - 3-Tier architecture
  - N-Tier architecture

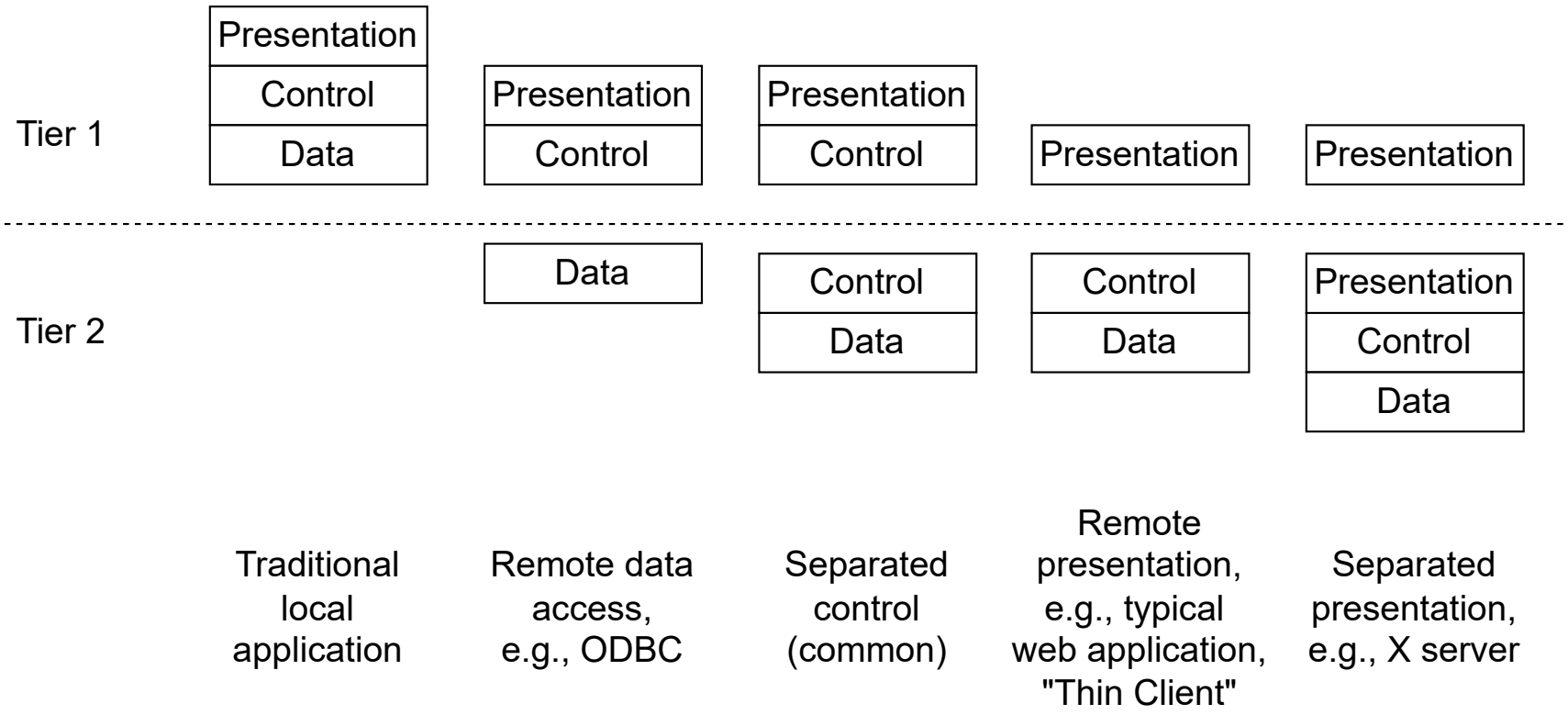
Two-Tier Architecture (1)

- Contains client tier (tier 1) and server tier (tier 2)
- Possible assignments



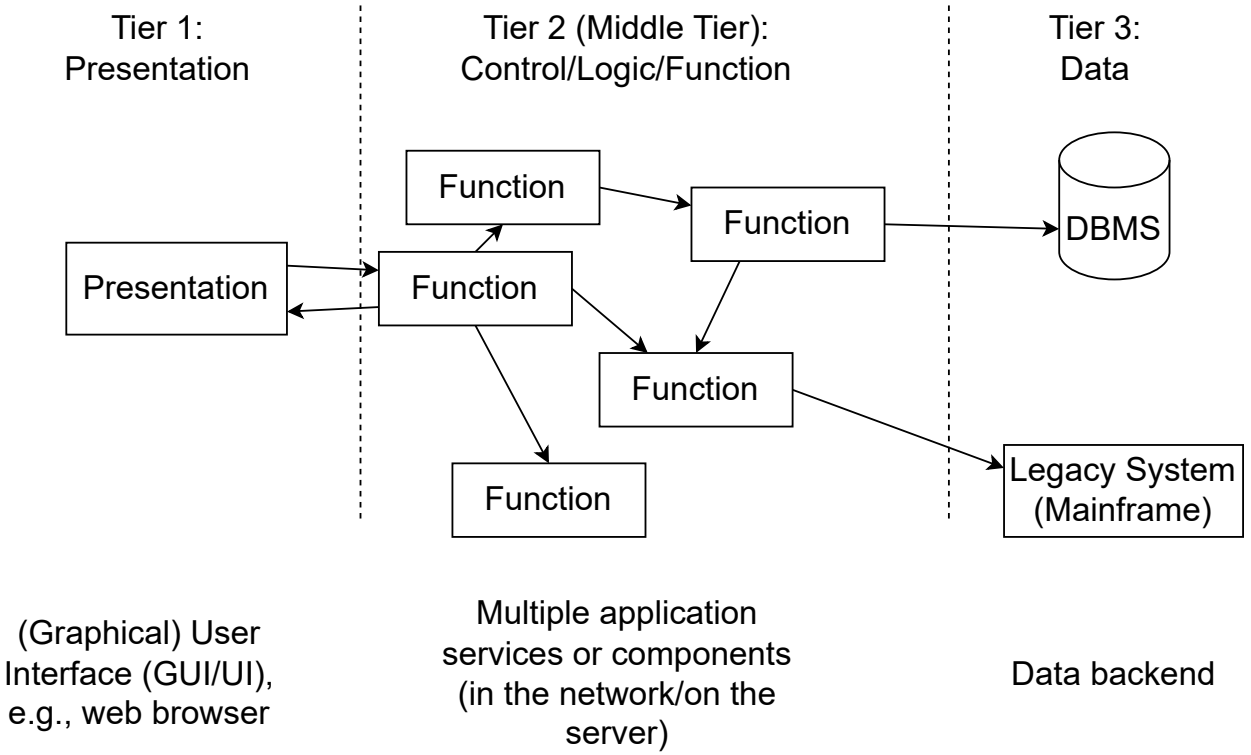
Two-Tier-Architektur (2)

- Different perspective



3-Tier Architecture

- Current structure model for complex applications



- Extension to N-Tier architecture
  Dividing primarily the middle tier

Example: J2EE Application

Client tier

- Internet browser or Java client

Web tier

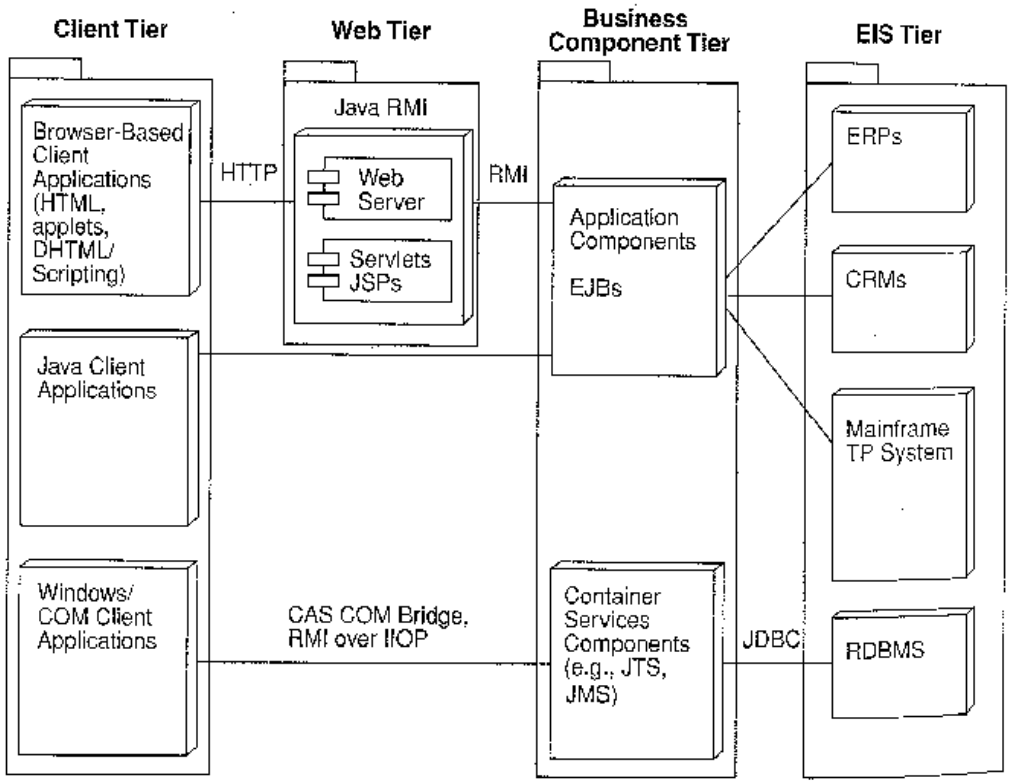
- Web server handling requests through JSPs and servlets

Business component tier (EJBs)

- Functional units that implement business rules and manipulate data

Enterprise information systems tier

- Databases, CRMs, mainframes, etc.



Source: R. Greespan

Important takeaway messages of this chapter

- Middleware acts as a layer between the OS and the application in order to abstract distributed applications from the underlying layers
- Middleware architectures describe the distributable units and interaction models
- For the design of a distributed system various architecture models can be used