



# Hauptseminar

## Management von Softwaresystemen

Techniken der System-Integration  
EAI, Middleware, SOA, CORBA

Betreuerin: Ulrike Hammerschall  
Referent: Alexey Krivoborodov

# Agenda

- Motivation
- Arten der Verteilung
- Vorteile der Verteilung
- Anforderungen an Lösungen zur System-Integration
- Lösungen:
  - Middleware
  - Enterprise Application Integration
  - Service-Oriented Architecture
  - Web Services
  - CORBA
- Zusammenfassung

# Motivation

- Wozu braucht man System-Integration?
- Viele Arten von verteilten Anwendungen
  - Einfache verteilte Anwendungen (WWW, FTP, E-Mail)
  - Verteilte Informationssysteme (Online-Banking, Flugreservierung)
  - Eingebettete verteilte Systeme (bis zu 70 Steuergeräten im Auto)
  - Mobile verteilte Systeme (mobile Telefone, PDAs)
- In vielen Unternehmen eine bunte Anwendungslandschaft von unabhängig agierenden Anwendungen
  - „Alte“ COBOL-Anwendungen am Mainframe
  - „Neue“ J2EE-Anwendungen auf einem Client-Server-System

# Warum Verteilung?

- Vorteile der Verteilung
  - Gemeinsame Nutzung von Hardware
    - Kostenersparnis
  - Gemeinsame Nutzung von Daten und Informationen
    - Informationen einer großen Anzahl von Usern zur Verfügung stellen
  - Gemeinsame Nutzung von Funktionalität
    - Zeitersparnis
    - Wiederverwendung

# Arten der Verteilung

- **Verschiedene Szenarien der Verteilung:**
  - **Anwendungskomponenten innerhalb einer verteilten Anwendung**
    - Hohe Abhängigkeit von Sender und Empfänger
    - Vorkommen unterschiedlicher Technologien eher unwahrscheinlich
    - Beispiel: Client/Server-Systeme
  - **Vollständige Anwendungen**
    - Niedriger Koppelungsgrad
    - Vorkommen unterschiedlicher Technologien wahrscheinlich
    - Beispiel: Business-to-Business-Systeme (B2B)
    - Beispiel: COBOL-Anwendung und eine J2EE-Anwendung
- **Aus verschiedenen Szenarien leiten sich verschiedene Anforderungen ab**

# Anforderungen an Lösungen zur System-Integration

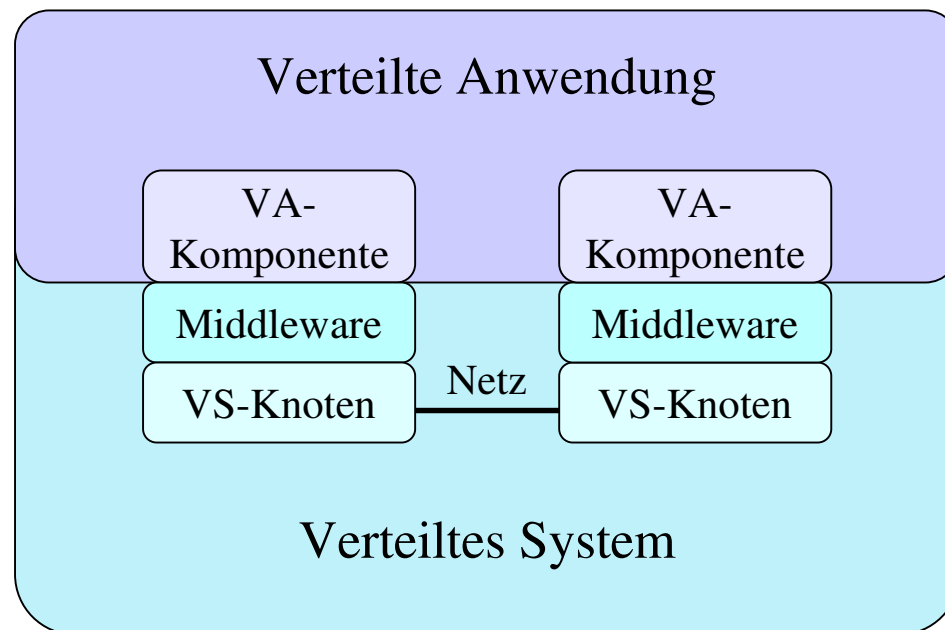
- Kommunikationsprotokoll
- Marshalling, Unmarshalling
- Fehlerbehandlung/Fehlerbehebung
- Aufgaben der Laufzeitumgebung
- Dienste

# Integrationsmethoden

- Integration über die Benutzungsschnittstelle
  - Auf bestehende Dialoge werden neue Dialoge aufgesetzt
- Integration über Funktionsaufrufe
  - Aus einem Anwendungskern wird die Funktionalität eines anderen Anwendungskerns aufgerufen
- Integration über Datenbanken
  - Mehrere Anwendungen kommunizieren über eine gemeinsame Datenbasis
- Integration über Komponenten
  - Plug-ins (Integration an dafür vorgesehenen Stellen)
  - Container (z.B. J2EE-Container für Java Beans (EJBs))

# Middleware

- Middleware ist eine intelligente Programmierschnittstelle, die zwischen verteiltem System und verteilter Anwendung liegt



Quelle: Hammershall [1]

# Charakteristika der Middleware

- Aufgaben der Middleware:
  - Verbergen der Aspekte der Netzwerkprogrammierung
  - Unterstützung der verteilten Anwendung in ihren Abläufen
- Kommunikationsorientierte Middleware
  - Abstraktion von der Netzwerkprogrammierung
- Anwendungsorientierte Middleware
  - Abstraktion von der Netzwerkprogrammierung und
  - Unterstützung der verteilten Anwendungen

# Kommunikationsorientierte Middleware

- Aufgaben
  - Kommunikation (Middleware-Protokoll)
  - Marshalling, Unmarshalling
  - Fehlerbehandlung/Fehlerbehebung
- Programmiermodelle
  - Entfernter Prozeduraufruf (RPC)
  - Entfernte Methodenaufrufe (RMI)
  - Nachrichtenorientiertes Modell
- Middleware-Technologien
  - Entfernte Aufrufe (Java RMI, Web Services/SOAP)
  - Nachrichtenorientierte Middleware (IBM WebSphereMQ - MQSeries)

# Anwendungsorientierte Middleware

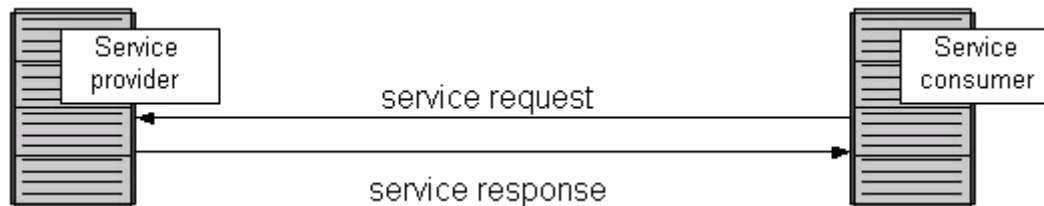
- Aufgaben
  - Verbindungsverwaltung
  - Verfügbarkeit
  - Sicherheit
- Dienste
  - Namensdienst
  - Sitzungsverwaltung
  - Transaktionsverwaltung
- Middleware-Technologien
  - Object Request Broker (ORB)
  - Application Server
  - Middleware-Plattformen (J2EE, .NET, CORBA)

# Enterprise Application Integration

- Vielfältige Anwendungslandschaft
- Strenge Anforderungen an Verfügbarkeit und Ausfallsicherheit bestehender Software
- Man muss berücksichtigen:
  - Heterogenität der Programmiersprachen
  - Heterogenität der Hardware und des Betriebssystems
  - Unterschiedliche Technologien
- EAI bezeichnet eine Mischung aus Konzepten, Technologien und Werkzeugen, die in ihrer Gesamtheit die Integration heterogener Anwendungen unterstützen
- Unterschied zu Middleware:
  - Middleware – Kommunikation zwischen Anwendungskomponenten
  - EAI – Integration eigenständiger Anwendungen

# Service-Oriented Architectures

- Ein Service ist eine Funktion:
  - mit einer wohldefinierten Schnittstelle
  - in sich geschlossen
  - unabhängig vom Kontext oder Zustand anderer Services
- Service-Oriented Architecture definiert eine Sammlung von Services
- Services kommunizieren miteinander
- Services können gemeinsam eine Aufgabe lösen

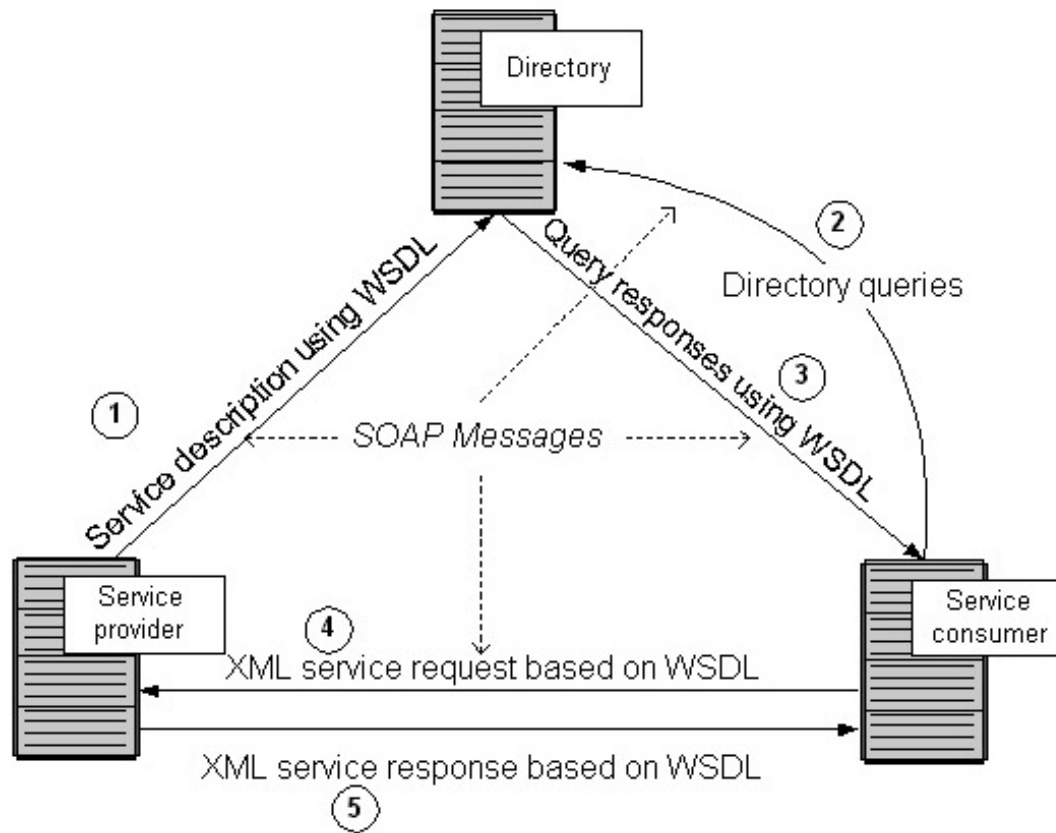


Quelle: <http://www.service-architecture.com/>

# Web Services

- Eine Middleware-Technologie für verteilte Anwendungen, vorzugsweise im Internet
- Datenübertragung per HTTP, SMTP
- SOAP als Protokoll des web-basierten entfernten Prozeduraufrufes
- UDDI als Verzeichnisdienst zur Registrierung von Web Services. Es ermöglicht das dynamische Finden des Web Services durch den Konsumenten
- WSDL zur Beschreibung der unterstützten Methoden und deren Parametern für den Programmierer

# Web Services



Quelle: <http://www.service-architecture.com/>

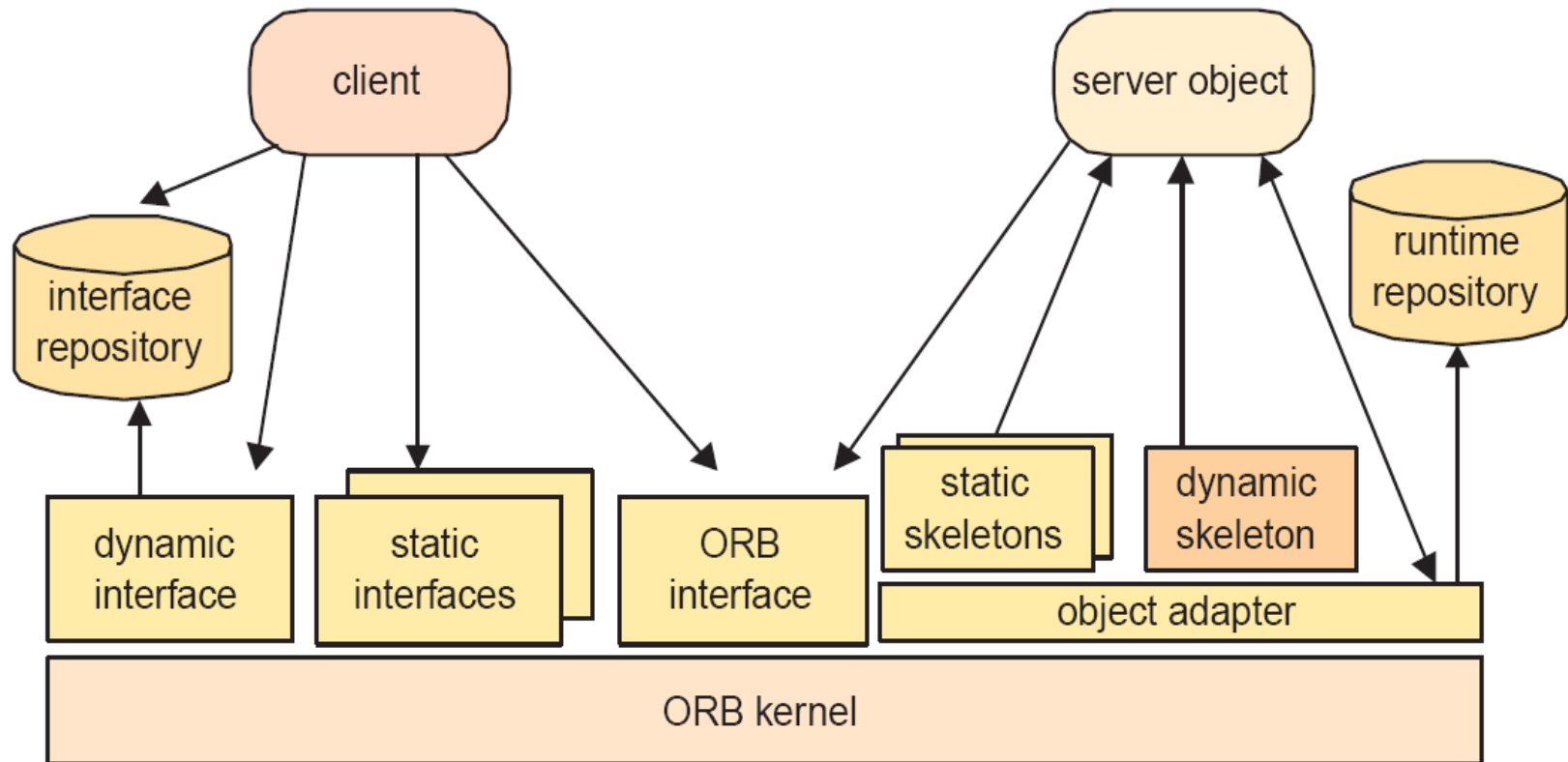
# Vorteile und Nachteile von Web Services

- Vorteile:
  - Unterstützung von offenen Standards (SOAP, WSDL, UDDI)
  - Keine Probleme mit Firewalls im Unterschied zu Java RMI
  - Eine offene und flexible Architektur, die unabhängig von den verwendeten Plattformen, Programmiersprachen und Protokollen ist
- Nachteile
  - Keine Übereinstimmung über Sicherheitsstandards und Transaktionskonzept
  - Performance (Overhead durch XML, Parsen, große Dateigrößen)
  - Bezahlen von kostenpflichtigen Diensten

# Common Object Request Broker Architecture (CORBA)

- Ein offener Middleware-Standard für verteilte Object Request Broker-Plattformen
- Ziel – die Bereitstellung einer Kommunikations- und Dienstinfrastruktur für verteilte Anwendungen
- Verantwortliches Gremium – die Object Management Group (OMG)
- Konkrete Ziele von CORBA:
  - Objektorientierung
  - Ortstransparenz
  - Sprachunabhängigkeit
  - Interoperabilität
  - Portabilität

# Architektur von CORBA



Quelle: Prof. Schlichter [6]

# Eigenschaften von CORBA

- CORBA services
  - Namensdienst (Zuordnung von Namen zu Objektreferenzen)
  - Transaktionsdienst (verteilte Transaktionen in objektorientierten Systemen)
- Einsatz
  - Ursprünglich als vollständige Plattform konzipiert
  - Reine Broker-Architektur zur objektorientierten Kommunikation durchgesetzt
  - Im kommerziellen Bereich in komplexe Middleware-Plattformen (J2SE, J2EE) integriert

# Zusammenfassung

- Verteilung ermöglicht die gemeinsame Nutzung von Ressourcen wie Hardware, Daten, Informationen und Funktionalität
- EAI fokussiert die Integration eigenständiger Anwendungen, Middleware konzentriert sich stärker auf die Kommunikation zwischen Anwendungs-komponenten
- Durch die Verwendung von Web Services entsteht eine offene und flexible Architektur, die unabhängig von verwendeten Plattformen, Programmiersprachen und Protokollen ist
- CORBA-Implementierungen finden heute eher selten als vollständige Middleware-Technologien Verwendung. Die ORB-Infrastruktur hat sich jedoch als tragfähig erwiesen und wird in vielen Fällen als Kommunikationsinfrastruktur eingesetzt

# Literatur

1. Ulrike Hammerschall: Verteilte Systeme und Anwendungen, Architekturkonzepte, Standards und Middleware-Technologien; Pearson Studium, 2005
2. Wolfgang Keller: Enterprise Application Integration, Erfahrungen aus der Praxis; Heidelberg, dpunkt Verlag, 1. Auflage 2002
3. William A. Ruh, Francis X. Maginnis, William J. Brown: Enterprise Application Integration, A Wiley Tech Brief; New York, NY, John Wiley & Sons, 2001
4. Andrew S. Tanenbaum, Maarten Van Steen: Distributed Systems: Principles and Paradigms; Prentice-Hall, 2002
5. Web Services and Service-Oriented Architectures  
<http://www.service-architecture.com/>
6. Prof. Dr. Johann Schlichter. Vorlesungsskript „Verteilte Anwendungen“ SS2005