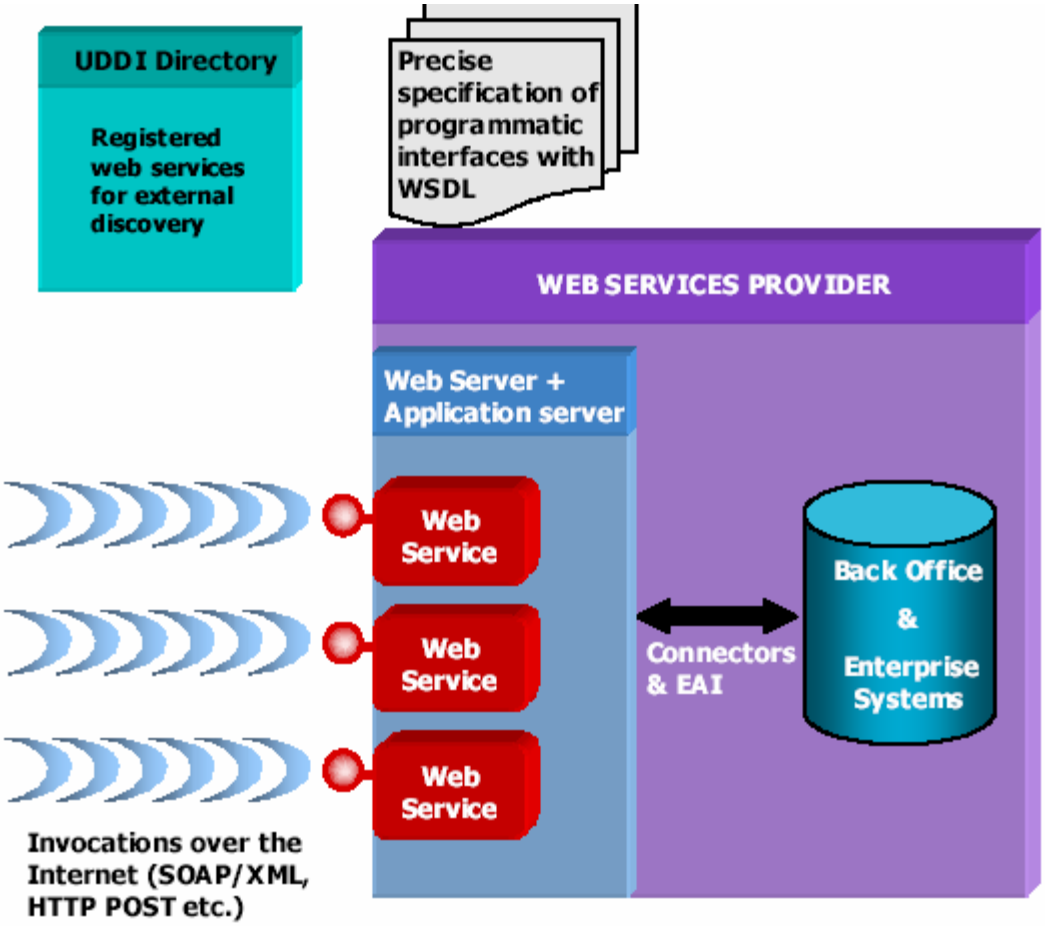
Vorlesung
Software aus Komponenten

3. Komponenten-Modelle

Prof. Dr. Hans-Gert Gräbe
Wintersemester 2005/06

Überblick über Webservice-Komponenten



XML-Protokolle

- **SOAP**
  Datenaustausch-format
- **WSDL**
  Beschreibungs-format
- **UDDI**
  Gelbe Seiten, Repositoryformat

Definition: SOAP

- **S**imple **O**bject **A**ccess **P**rotocol
- Entwickelt von Microsoft, DevelopMentor, UserLand
- Seit 2000 W3C-Spezifikation
- Standardisiertes Verpackungsprotokoll auf XML-Basis für Nachrichtenaustausch zwischen Anwendungen
- Stellt die Message-Spezifikation zur Kommunikation von Web-Services dar
  - einfacher XML-Umschlag um die zu übertragende Information + Regeln zur Darstellung anwendungs- und plattform-spezifischer Datentypen in XML
- Applikationen können so über das Internet Daten und Dokumente austauschen

Nachrichtenaustausch

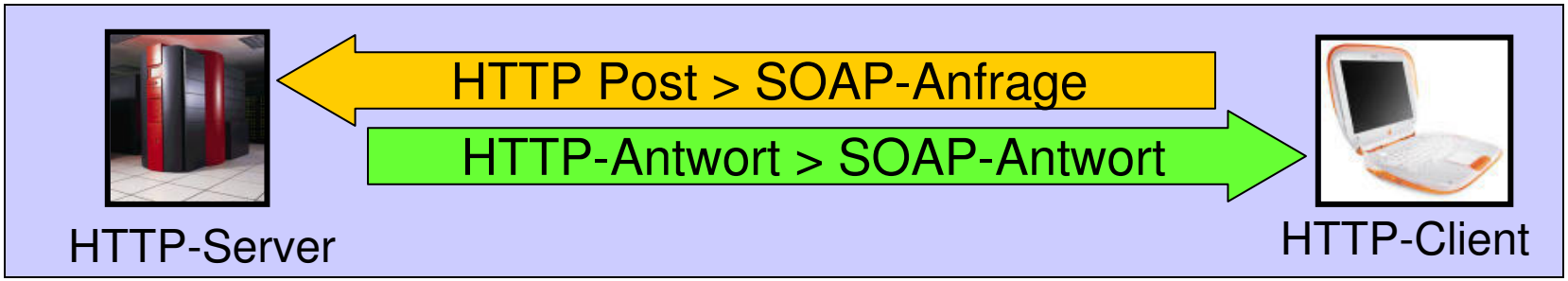
- Anwendungen können Nachrichten (z.B. Aktienkurse, Warenbestellungen) in XML codieren und via SOAP-Dokumenten austauschen
- Durch XML liefert SOAP eine **plattformunabhängiges** Protokoll für den Nachrichtenaustausch
- SOAP stellt Konventionen für eine **standardisierte Darstellungsweise** der Informationen in XML für den Datenaustausch in heterogenen Systemen zur Verfügung
- SOAP liefert zwei Ansätze für den Nachrichtenaustausch:
  - ***Remote Procedure Calls***
    Funktionsaufrufe entfernter Prozeduren im Sinne verteilter Architekturen
  - ***Electronic Document Interchange***
    Dokument-basiertes SOAP, bei dem fachliche Dokumente z.B. Steuererklärung, Warenbestellungen ausgetauscht werden

SOAP-Nachrichten

- SOAP-Nachrichten bestehen aus einem Umschlag (envelope) mit den Unterelementen:
  - Kopf (header, optional)
    - kann mehrere Header-Blöcke enthalten
  - Rumpf (body, erforderlich)
    - Jede Nachricht enthält **genau einen** Rumpf
- Der Kopf enthält Informationen:
  - Wie die Nachricht verarbeitet werden soll
  - Routinginformationen
  - Authentifizierung, Autorisierung und Transaktionskonzepte
  - Sonstige Kontextinformation
- Der Rumpf enthält die Nachricht bzw. Funktionsaufrufe in XML-Syntax (<tag>...</tag>)
- Die XML-Syntax von SOAP 1.2 basiert auf dem Namensraum http://www.w3.org/2001/06/soap-envelope

Transport der Nachrichten

- SOAP ist als Verpackungsprotokoll konzipiert und ist deswegen entkoppelt von der Netzwerk- oder Transportschicht
- SOAP kann so flexibel, je nach Anspruch und Einsatzanforderung, durch beliebige Transportprotokolle, wie HTTP, FTP, TCP, SMTP, POP3, MQSeries oder Jabber verschickt werden.
- Das am häufigsten verwendete Transportprotokoll ist HTTP
- Die SOAP-Spezifikation geht gesondert auf SOAP-über-HTTP ein und beschreibt wie der SOAP-Nachrichtenaustausch mittels HTTP realisiert werden kann
- SOAP eignet sich besonders in der RPC-Form für HTTP, da HTTP ebenso mit Requests und Responses arbeitet



Transport der Nachrichten (2)

Beispiel eines SOAP-Requests über HTTP:

```
POST /soapworkshop/services/id/id.asp HTTP/1.1
Host: xxx.xxx.xxx.xxx
Content-Type: text/xml
Content-Length: nnn
SOAPAction: „urn:myDatabase#GetPersonID"

<?xml version="1.0"?>
<S:Envelope xmlns:S=,...'>
  <S:Body>
    <vb:GetPersonID xmlns:vb=,...'>
      <person>Baron Münchhausen</person>
    </vb:GetPersonID >
  </S:Body>
</S:Envelope>
```

Der HTTP-Header **SOAPAction** wird in der SOAP-Spezifikation definiert.

SOAP-Action zeigt dem HTTP-Server, **was die SOAP-Nachricht will**, bevor der XML-Code entschlüsselt wird.

Definition: WSDL

- **W**eb-**S**ervice **D**escription **L**anguage
- eine Art IDL in XML für Web-Services
- Standard des W3C
- entwickelt in 2001 von Ariba, IBM, Microsoft
- beschreibt Web-Services als eine Menge von Endpoints, die mit **Messages** arbeiten, welche dokumenten-orientiert oder prozedur-orientiert sind
- Operationen und Messages werden **erst** abstrakt beschrieben und **dann** an das jeweilige Netzwerk-Protokoll und Message-Format gebunden (SOAP, HTTP, MIME)

Beschreibung eines Webservice

- WSDL ermöglicht die automatisierte Erstellung von Clients für Webservices
- Die Operationen von Webservice (SOAP-RPC) setzen voraus, dass korrekte Funktionsaufrufe oder fachlich korrekte Dokumente ausgetauscht werden.
- WSDL stellt die Schnittstellenbeschreibung für diese Operationen oder Dokumente zur Verfügung
- WSDL beschreibt die Implementierung und die von Webservice-Konsumenten zu nutzenden Protokolle (Verpackung, Transport) für den Informationsaustausch
- WSDL dient dem Webservice dazu, standardisiert bekannt zu geben, welche Funktionen er besitzt, wie die Schnittstellen (Parameter) aussehen und über welche Protokolle Konsumenten ihn ansprechen können.

Beschreibung eines Webservice (2)

Vorteile von WSDL:

- Entwicklung und Pflege von Diensten wird über das standardisierte Schnittstellenformat WSDL einfacher
- Die Nutzung von Webservices kann automatisierter geschehen und Fehlerquellen bei der Clienterstellung für den Webservice-Zugriff werden minimiert
- WSDL erlaubt die dynamische Entdeckung und Anpassung der Clients an implemetierte Änderungen der Webservice-Funktionen
- WSDL lässt sich über Entwicklertools (z.B. MS .NET) automatisch aus den Applikationskomponenten erstellen

Beschreibung eines Webservice (3)

Notwendigkeit und Aufgabenbereich von WSDL:

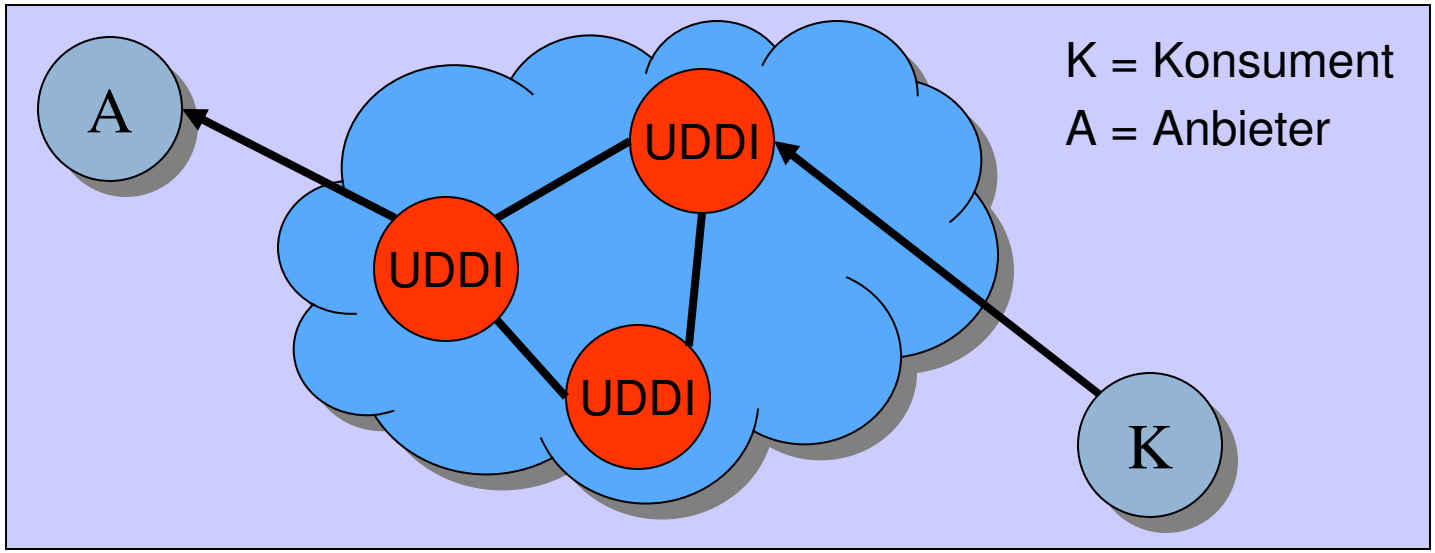
- WSDL ist nicht zwingend für die Kommunikation mit Webservices erforderlich.
- Wenn der Konsument weiß, wie mittels SOAP mit dem Webservice kommuniziert, ist WSDL unnötig.
- Bei dynamischen Webservice-Aufrufen, wo noch nicht bekannt ist, wie mit dem unbekannten Dienst kommuniziert werden soll, stellt WSDL die Informationen für diese Kommunikation bereit

Definition: UDDI

- **U**niversal **D**escription, **D**iscovery and **I**ntegration
- „Gelbe Seiten" für Web-Services, damit ein Dienstkonsument die Webservices auch findet, um diese zu benutzen.
- dient zur Lokalisierung und Veröffentlichung von Web-Services im Internet
- UDDI = Register für Dienste und ihre Beschreibungen + Suchmethoden + Veröffentlichungsmethoden
- UDDI-Daten enthalten Kontakt-Informationen, Listen von Business Services und Infos, wie ein Service via Protokoll angesprochen werden kann
- Seit Sept. 2000 von Microsoft, Ariba, IBM und 33 anderen entwickelt, mehr als 300 Mitgliedsorganisationen
- ***„Making it possible for organizations to quickly discover the right business from millions currently online" (uddi.org)***

Definition: UDDI (2)

- UDDI besteht aus einem **Netzwerk** miteinander verbundener Registries
- Alle UDDI-Registries haben mit SOAP alle dieselbe Webservice-Schnittstelle für **Veröffentlichen** und **Finden** für Webservices implementiert.
- Es existieren auch **private** UDDIS (Firmen-UDDIs), um private Services und Methoden über das Internet verfügbar zu machen und in die verteilte Unternehmenslandschaft zu integrieren.



Erste Realisierungen des Komponentenkonzepts

- Der dokumentenzentrierte Ansatz
  - Zusammenstellen von Dokumenten aus Teilen mit unterschiedlicher Funktionalität (Tabellenkalkulation, Grafiken)
  - Beispiel: MS-Word, Hypercard
- Der webzentrierte Ansatz
  - Anreicherung von Webseiten mit funktionalen Komponenten (Abfrageteile, Applets)
- Der geschäftsprozesszentrierte Ansatz
  - Integration von Applikationen verschiedener Hersteller in einer einheitlichen betrieblichen Infrastruktur

Erste Realisierung des Komponentenkonzepts
Der dokumentenzentrierte Ansatz

- Idee: Nutzer wird nicht mit vielen verschiedenen Applikationen konfrontiert, sondern mit Dokumenten, die aus mehreren Teilen bestehen können. Diese Teile können unterschiedliche Applikationen zur Darstellung benötigen, kennen diese aber selbst.

- Erste Realisierung unmittelbar auf der Ebene von integrierten Textdokumenten
  - Hypercard (Apple)
  - Word mit Visual Basic und VBX (Microsoft)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Visual Basic

- Dokument besteht aus (mehreren) Formularen
- Formular kann mit Kontrolleinheit ausgestattet sein
- Kontrolleinheiten interagieren über Basic-Skripte

Flexibilität und Produktivität dieses Konzepts führten zur Herausbildung des ersten Komponentenmarkts mit Komponenten etwa zur Tabellenkalkulation oder zur Prozessautomatisierung.

OLE als Weiterentwicklung dieses Ansatzes

- Formulare -> Container für beliebige Anwendungen
- Kontrolleinheit -> Dokumentenserver
- Container können hierarchisch ineinander geschachtelt werden

Erste Realisierung des Komponentenkonzepts

Der webzentrierte Ansatz

- Idee: Einbettung von beliebigen Objekten in HTML-Seiten
  - z.B. Java Applets, Form-Bestandteile
- Einheitliche und erweiterbare Darstellung im Browser durch Plugin-Technologie
- Schritt weg vom OLE-Containerkonzept und zurück zum (nicht hierarchischen) Formularansatz von Visual Basic

Aktuelle Entwicklungsrichtungen

- COM (Microsoft)
- CORBA (Object Management Group)
- Java (Sun und inzwischen auch IBM)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Die OMG und CORBA

- Geschichte, Zielstellungen, Entwicklungsetappen
- Architektur
  - Objekte, Servanten, Anwendungen
  - Schnittstellensprache OMG IDL
  - Dynamische Methodenaufrufe (DII)
  - Symmetrie des CORBA-Modells
- Der Object Request Broker (ORB)
- CORBA-Objekte und Objektreferenzen
- CORBA IDL und Datentypen

- Literatur: CORBA Spezifikation 3.0 (Juli 2002)
  - 1154 Seiten pdf-Dokument, siehe http://www.omg.org

Zur Geschichte der OMG (Object Management Group)

- **Ausgangspunkt 1989:** Wie kommuniziert man in einem verteilten OO-System über Sprach- und Plattformgrenzen hinweg?
  - selbst auf derselben Plattform lieferten C++-Compiler inkompatiblen Bytecode, verschiedene Objektmodelle in verschiedenen Programmiersprachen, Plattformunterschiede bei Socket-Kopplung
  - „Deep gaps everywhere"
- im April 1989 von 11 Firmen gegründet
- heute mit ca. 800 Mitgliedern eines der größten Konsortien der Computer-Industrie
  - vor allem Systemanbieter und Anwender objektorientierter Techniken

Zielstellung: „Standardisierung, koste es, was es wolle", um Interoperabilität auf allen Ebenen in einem offenen Markt für „Objekte" zu erreichen.

Zielstellungen der OMG

- Offene Interoperabilität zwischen einer Vielzahl von Sprachen, Implementierungen und Plattformen
- mehr standardisieren als „binäre" Standards
- Flexibilität statt Binärkompatibilität
  - „teure" Hochsprachenprotokolle
- **Nichtkommerzielle Vereinigung** zur Entwicklung von technisch exakten und in der Praxis realisierbaren Spezifikationen
- **Vereinbarung von Standards und Spezifikationen** der Infrastruktur für verteilte, objektorientierte Anwendungen
- **Aufstellung von Richtlinien** (guidelines) zur Entwicklung von Umgebungen, in denen heterogene Systemen (verschiedene Plattformen, Betriebssysteme u.ä.) zusammenarbeiten können
- Durch standardisierte, objektorientierte Softwarekonzepte die **Entstehung eines Marktes für Komponentensoftware forcieren**

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Etappen der Entwicklung von CORBA

CORBA 1 (seit 1991) : **Standardisierung des ORB**

- erste Lösungen, um das Wirrwar zu entflechten
- Ansatz: Vermittlung zwischen Anfragen und Diensten durch einen Object Request Broker (ORB)
- CORBA = Common Object Request Broker Architecture
- **Meilenstein**: Schnittstellen-Definitionssprache (OMG IDL)

CORBA 2 (seit 1995 – 96) : **Interoperationsstandards zwischen ORBs**

- **Meilenstein**: Internet Inter-ORB Protokoll (IIOP)
- muss von jeder ORB-Implementierung unterstützt werden
- Für CORBA 2 existiert Vielzahl von Realisierungen verschiedener Anbieter und für verschiedene Plattformen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

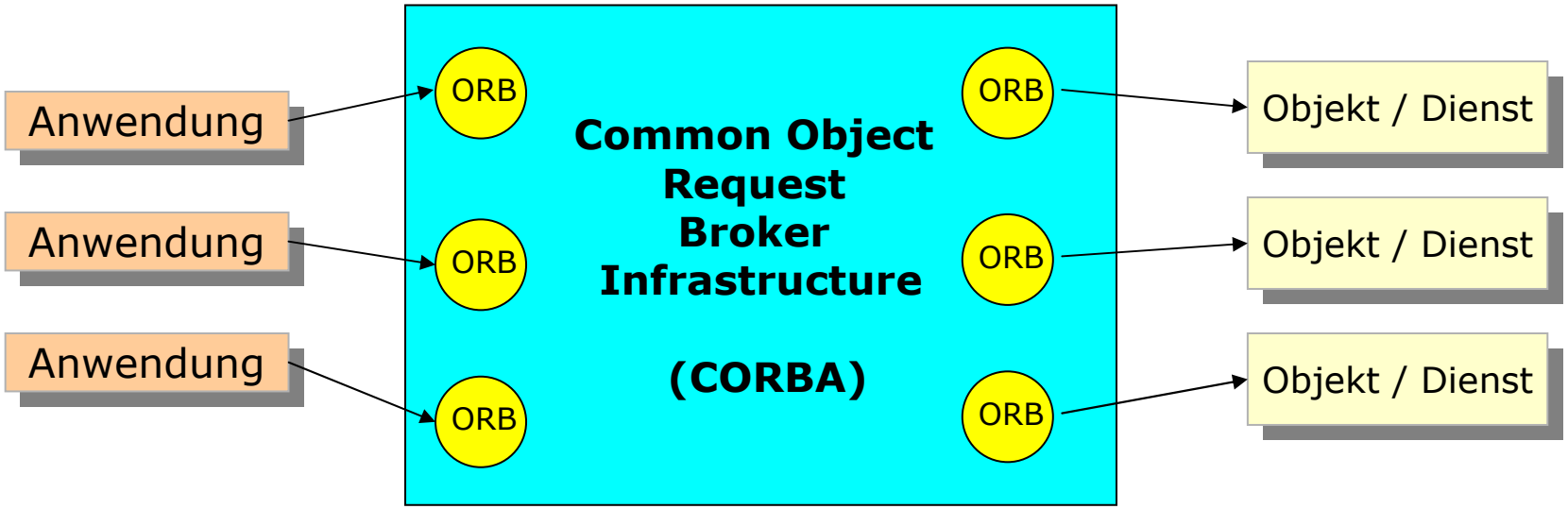
Etappen der Entwicklung von CORBA (Fortsetzung)

CORBA 3 (12/2002) : **Komponenten- und Systemintegration**

- Höhere Abstraktionsebene
- neue Sprachebenen zur Beschreibung von Komponenten-Eigenschaften
- seit 1998 in der Entwicklung, aber als Ganzes erst Ende 2002 freigegeben
  - CORBA 2.3 ... 2.6 (2001) : Freigabe verschiedener Standards, auf die man sich auf dem Weg zu CORBA 3 zwischenzeitlich geeinigt hatte
- **Meilenstein**: CORBA Komponentenmodel (CCM)
  - Version 3.0, Juli 2002
- aktuelle Version CORBA/IIOP 3.0.3, März 2004 (http://www.omg.org)
- bisher kaum Implementierungen, die CORBA 3 voll unterstützen

CORBA besteht im Grunde aus drei wichtigen Teilen:

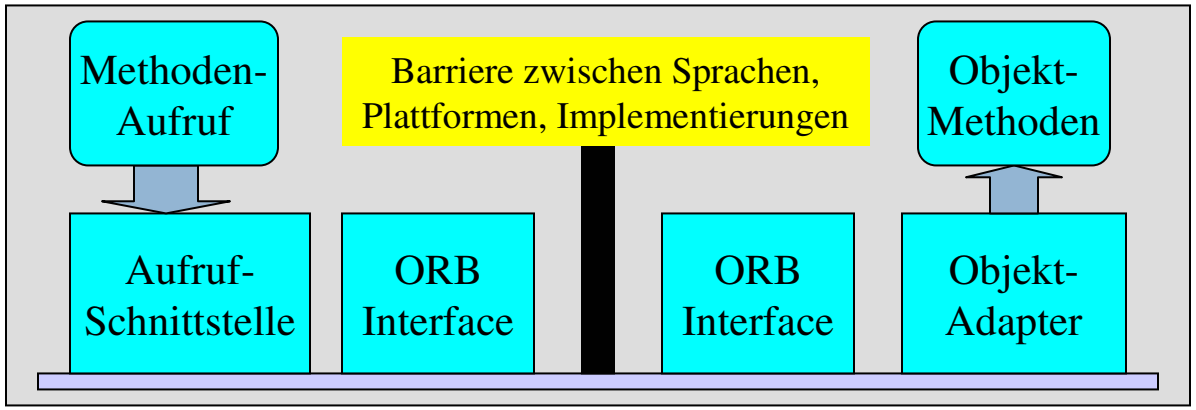
- einer Menge von Aufrufschnittstellen (Invocation Interfaces)
- den Vermittlern (Object Request Brokers – ORBs) als den Schaltstellen der Kommunikation
  - mit einem spezifizierten Protokoll, dem internet inter-ORB protocol IIOP
- einer Menge von Objekt-Adaptern



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Laufzeitbindung von Methodenaufrufen

- Aufrufschnittstelle serialisiert Aufrufargumente
- ORBs suchen Zielobjekt, -methode, organisieren Transport der Argumente
- Objektadapter: dient der Aktivierung des Diensts im Objekt. Deserialisiert Argumente und ruft entsprechende Methode des Zielobjekts auf.

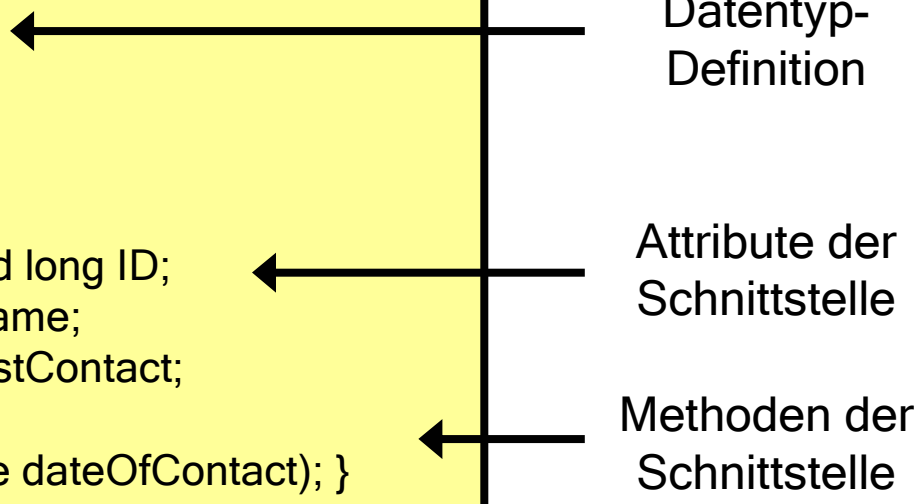


Wichtige Voraussetzungen

1. Schnittstellen müssen in einer einheitlichen Sprache **definiert** werden (**Interface Definition Language - OMG IDL**)
   - wesentlicher Bestandteil des CORBA-Standards
   - ermöglicht generisches Serialisieren / Deserialisieren

```
module Example {
  struct Date {
            unsigned short Day;
            unsigned short Month;
            unsigned short Year; }

interface Ufo {
            readonly attribute unsigned long ID;
            readonly attribute string Name;
            readonly attribute Date FirstContact;
            unsigned long Contacts ();
            void RegisterContact (Date dateOfContact); }
}
```



Wichtige Voraussetzungen (Fortsetzung)

2. alle Programmiersprachen, die den CORBA-Standard unterstützen, müssen **an OMG IDL gebunden** werden.
   - Mapping von Datentypen,
   - Übersetzung des OMG IDL Operationsformats in das sprachspezifische Aufruf-Format
   - Fehlerbehandlung
   - existieren Anbindungen für C, C++, SmallTalk, Cobol, Java, ...

In OMG IDL beschriebene Schnittstellen werden dann

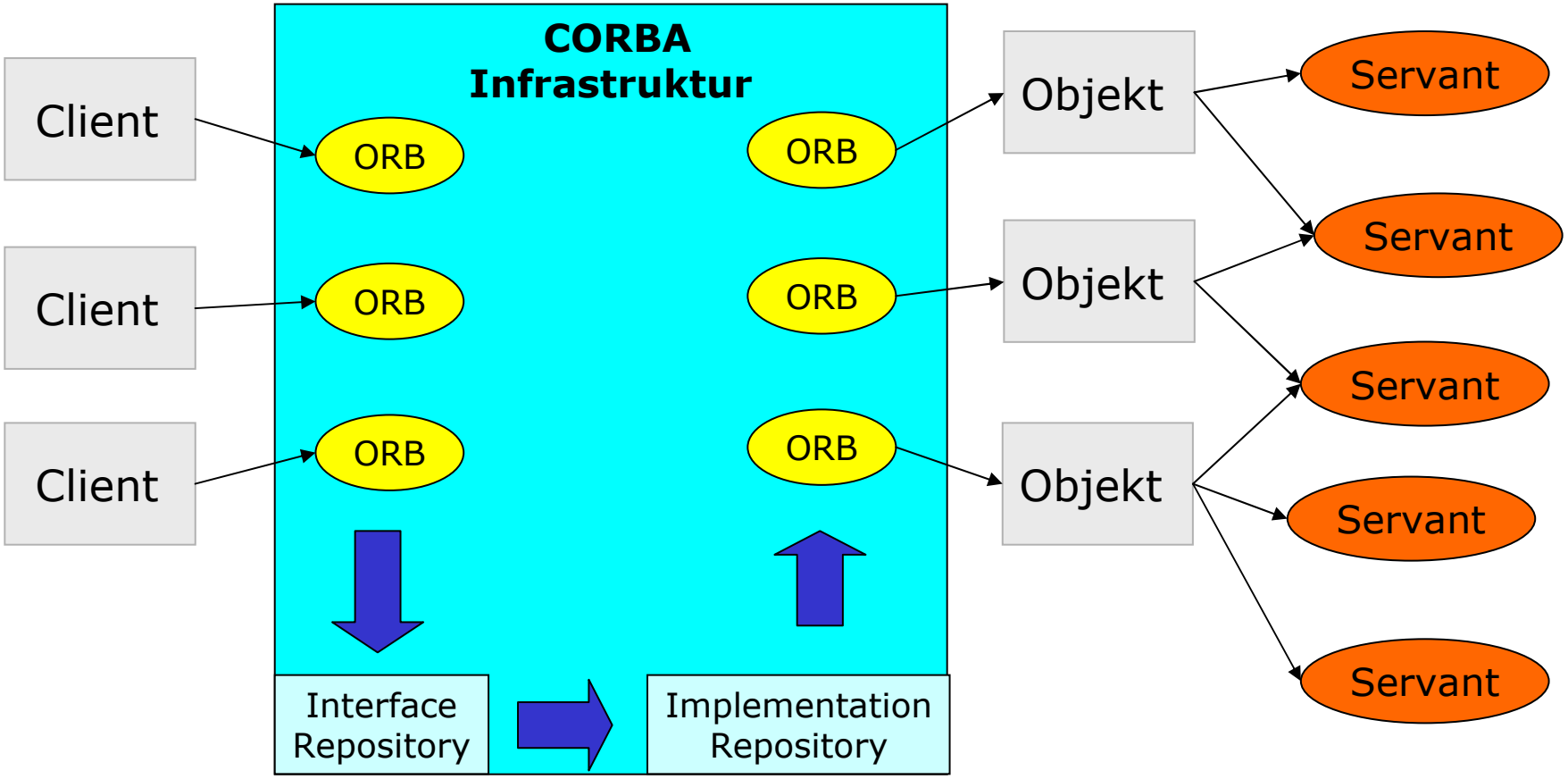
- mit einem **OMG IDL Compiler** übersetzt
- im **Schnittstellen-Repository des ORB** abgelegt
- durch **Methoden der ORB-Schnittstelle** angesprochen

Wichtige Voraussetzungen (Fortsetzung)

3. Programmfragmente stellen **Implementierungen** für solche Schnittstellen (oder Teile davon) bereit
   - heißen **Objekt-Servanten** (object servant)
   - werden im **ORB-Implementations-Repository** registriert
   - Servanten werden vom ORB bei Bedarf geladen und/oder gestartet
   - Objektadapter teilen dem ORB mit, welche Objekte von welchen Servanten bedient werden.
   - Eine Serverumgebung (typ. Prozess) kann mehrere Servanten bedienen.
   - *:* - Beziehung zwischen Objekten und Servanten

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Architektur im Überblick



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Stummel (stubs) und Skelette (skeletons)

- Methodenaufrufe erfolgen über Stummel-Skelett-Prinzip des RPC
  - mit OMG IDL Compiler aus Schnittstellenbeschreibung generierbar
- direkt nur für statische Methodenaufrufe einsetzbar (static invocation interface SII / static skeleton interface SSI)