

Client/Server-Systeme

Prof. Dr.-Ing. Wilhelm Spruth

SS 2003

Teil 14
CORBA, RMI, Web Services

Wiederverwendbarkeit von Code

Vorbild: Entwurf/Bau einer Brücke im Bauingenieurwesen

Objekttechnologie ermöglicht Code Blöcke mit fest definierter Funktionalität, die in Klassen gekapselt werden. Ein Objekt als Instanz einer Klasse stellt sich dem Entwickler als Black Box mit einer öffentlichen Schnittstelle dar.

Wird nur in einer einzigen Sprache mit identischem Compiler entwickelt ist die Wiederverwendbarkeit von Klassen am leichtesten erreichbar. Beim Einsatz mehrerer Programmiersprachen muß die Objektphilosophie der Programmiersprache beachtet werden. Z.B. unterstützt C++ die Mehrfachvererbung, Java aber nur die Einfach-vererbung.

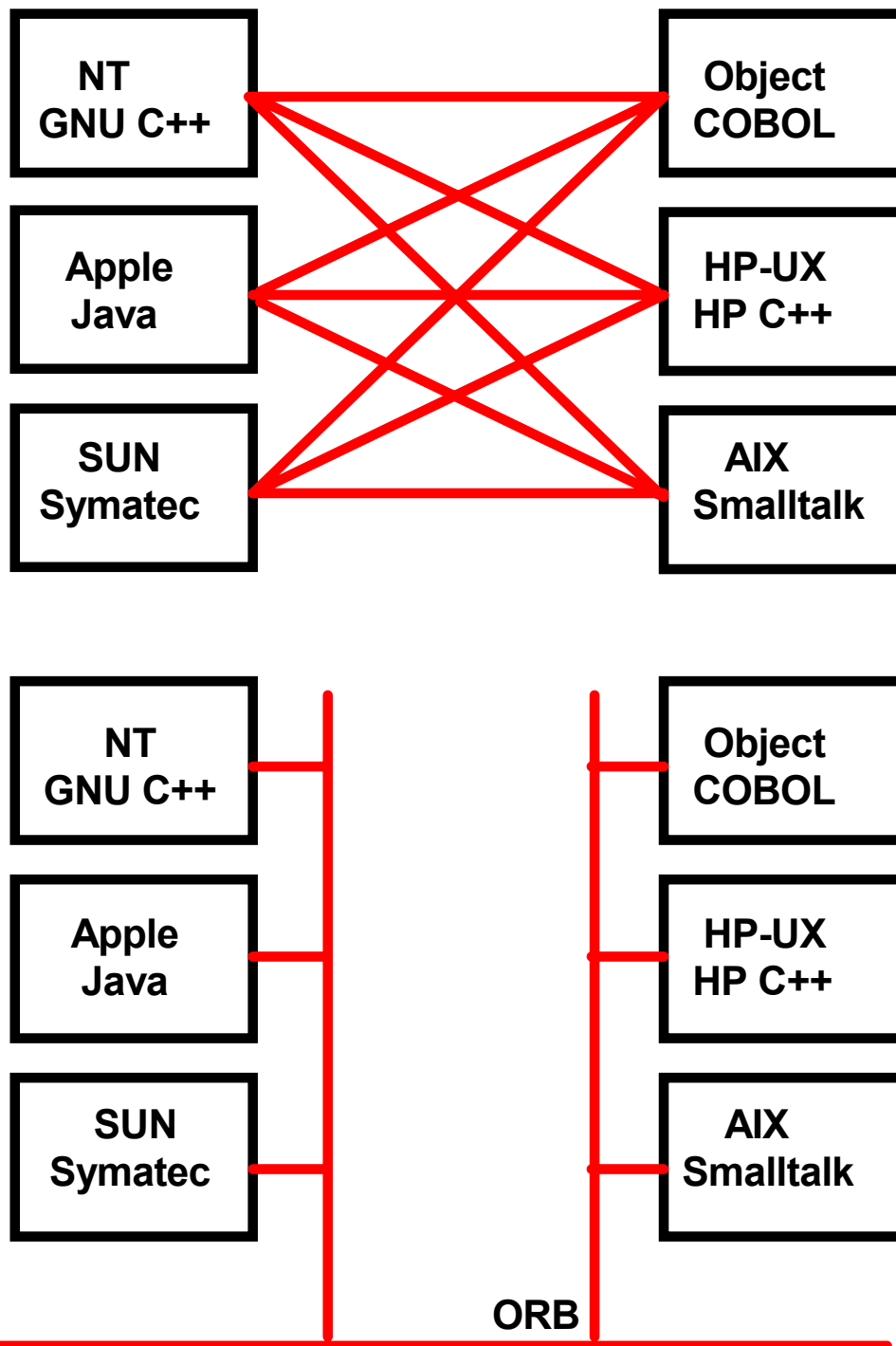
Um objektorientierte Bibliotheken mit unterschiedlichen Sprachen und Compilern zu realisieren, ergeben sich eine Reihe von Problemereichen:

- Sprachunabhängigkeit: Smalltalk- und C++-Objekte verstehen einander nicht
- Compilerunabhängigkeit: Objekte zweier Compiler (z.B. *Watcom* und *GNU C++*) können nicht miteinander kommunizieren, weil die Verwaltung interner Informationen nicht standardisiert ist
- starre Kopplung zwischen Objekten: Wenn sich die Implementierung einer Klasse ändert, müssen alle Teile, die diese Klasse in irgendeiner Form nutzen, neu kompiliert werden. Beispiel: C++ Compiler benutzen Konstanten beim Zugriff auf Daten und Methoden
- Beschränkung auf einen Prozeßraum (Objekte können nicht über Prozeßgrenzen hinweg kommunizieren)

Zielsetzung: Unterschiedliche objektorientierte Klienten-Implementierungen, die auf unterschiedlichen Plattformen (z.B. HP-UX, AIX, Solaris, Linux, NT, OS/400, OS/390) laufen, sollen mit unterschiedlichen objektorientierten Server-Implementierungen (ebenfalls unterschiedliche Plattformen) in Binärform kompatibel zusammen arbeiten.

Klienten sollen maximal portierbar sein, und ohne -Änderungen auf Rechnern/Betriebssystemen unterschiedlicher Hersteller lauffähig sein.

Klienten sollen keine Kenntnis benötigen wie ein Objekt auf einem Server implementiert ist



In unterschiedlichen Programmiersprachen entwickelte binäre Objekte können plattformunabhängig miteinander kommunizieren.

Object Request Broker ORB

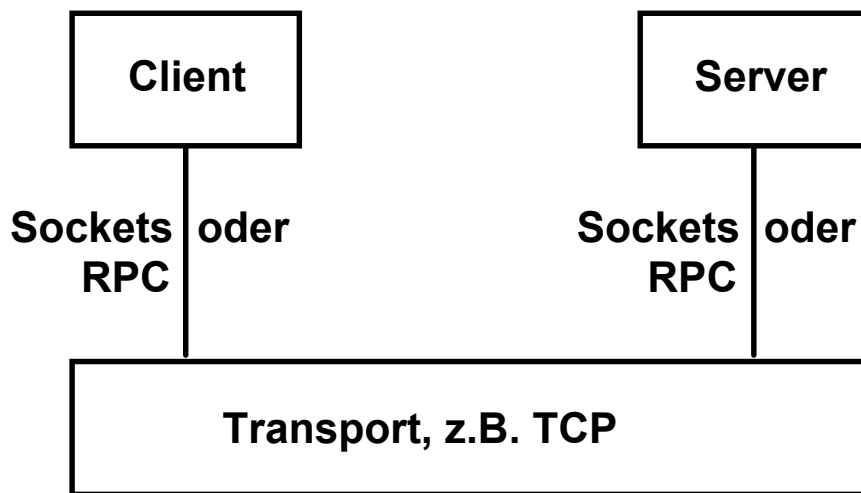
Ein ORB (Object Request Broker) ist eine Komponente eines Betriebssystems. Er ermöglicht es, daß Objekte, die in unterschiedlichen Programmiersprachen entwickelt wurden, in binärer Form miteinander zusammenarbeiten. Dies kann über Prozeß- und physische Rechengrenzen hinweg geschehen.

Hierfür stellt der ORB ein einheitliches Klassenmodell sowie Standards für die Organisation von Klassenbibliotheken und die Kommunikation zwischen Objekten zur Verfügung.

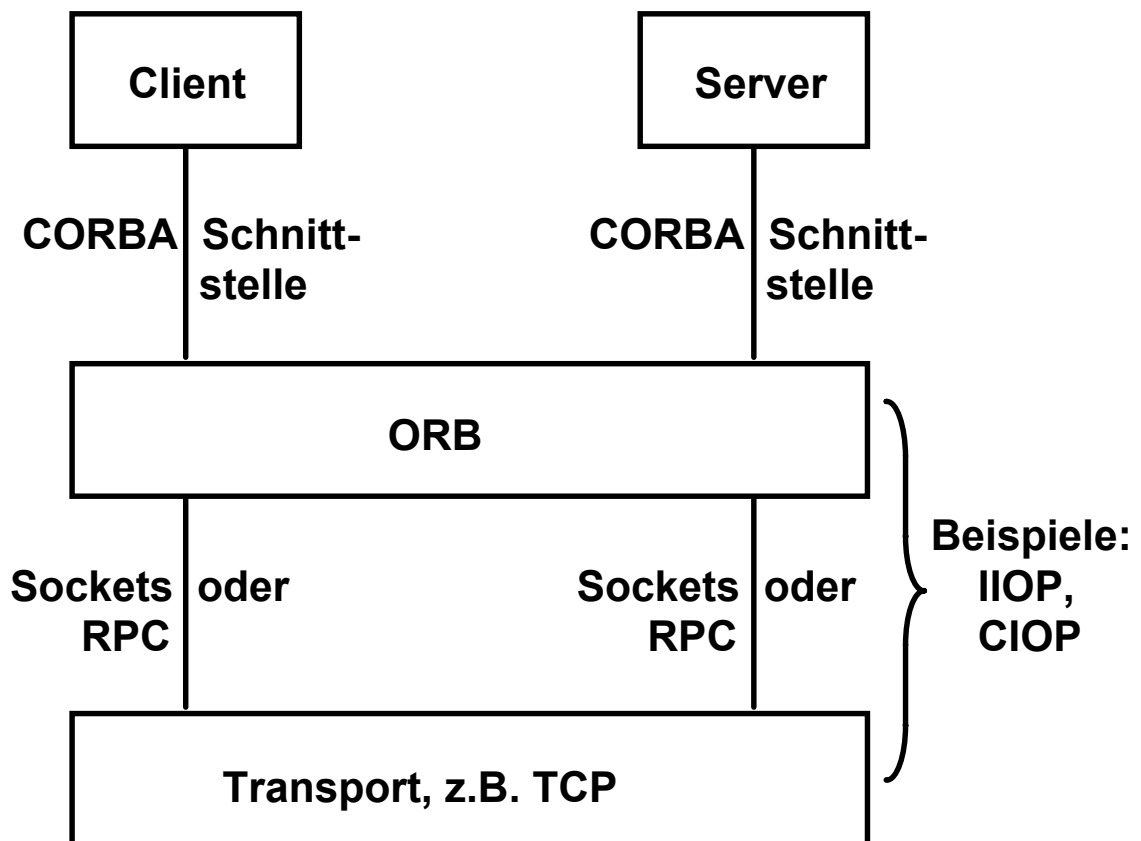
Es existieren mehrere Alternativen einen ORB zu implementieren:

- **CORBA Standard der OMG**
- **Remote Method Invocation (RMI),
 Teil des JDK 1.1 der Fa. SUN**
- **COM+ / DCOM / Activex / SNA / DotNet der Fa. Microsoft**

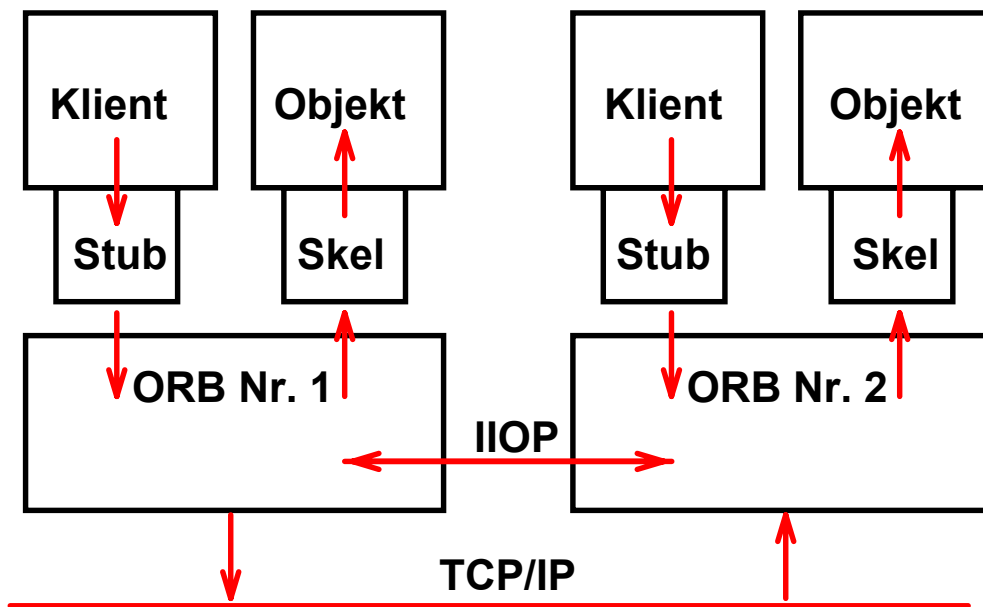
In eine ähnliche Richtung geht die Internet orientierte SOAP / UDDI / WSDL Initiative.



Mit Prozeduren arbeitendes Client/Server-System



Objektorientiertes Client/Server-System



ORB stellt eine Schnittstelle zwischen Client und Server dar, die ähnlich der Socket- oder RPC-Schnittstelle arbeitet, aber

- auf einer höheren Ebene arbeitet (und deshalb evtl. Sockets oder RPC für die eigentliche Kommunikation verwendet)
- objektorientiert arbeitet
- für die Kommunikation das IIOP Protokoll verwendet (andere)

Klienten und Server Objekte kommunizieren mit ihrem ORB über Stubs und Skeletons. Sie können transparent auf dem gleichen oder auf entfernten Rechnern arbeiten

Eine eindeutige Objekt Referenz (IOR) wird dem Objekt bei der Entstehung zugeordnet. Der Client nimmt die Dienste eines Objektes in Anspruch, indem er einen Methodenaufruf ausführt. Der Methodenaufruf enthält:

- IOR
- Methodennamen
- Parameter
- Kontext (enthält Information über die Position des Aufrufers und nimmt Fehlermeldungen entgegen)

Interoperable Object Reference IOR

Zeichenkette, die ein Objekt innerhalb eines verteilten CORBA Systems eindeutig kennzeichnet.

Vergleich:

<u>Internet</u>	<u>CORBA</u>
134.2.2.102	Object Reference
informatik.uni-tuebingen.de	String (Name)

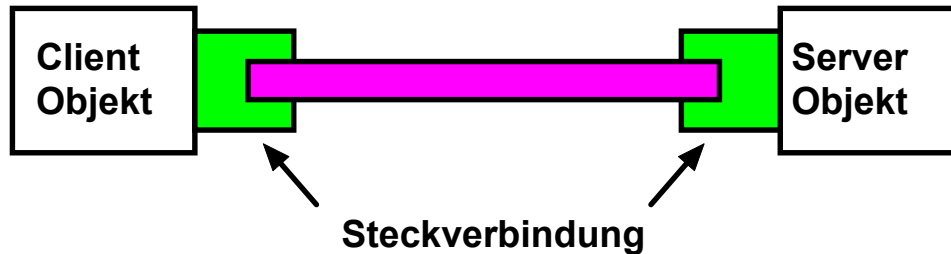
CORBA ORB's unterschiedlicher Hersteller übermitteln Object References nach dem IOR (Interoperable Object Reference) Standard.

Anwendungen verwenden String Namen. Es ist die Aufgabe eines Name Servers, String Namen in Object References zu übersetzen.

An Interoperable Object Reference

***IOR:0000000000000003249444c3a494e616d696e674d616e616
765644f626a656374434453496d706c2f4e616d696e67436f6e7
46578743a312e30000000000000010000000000000168010101
000d000000392e3136342e3138322e353700008403bc0000000
432333645303145332d373234442d313644442d454646332d30
3134393039413442363339000200005203000033000f0069424f
494d436f6e7461696e65720003000e00695472616e735379734f
626a73000e00536f6d436f6e7461696e65724900580016006943
44534e616d696e67436f6e74657874486f6d65003e002f2e2e2e2
f706f6f6c33345f63656c6c2f4342432d6c6f63616c2d726f6f747
32f706f6f6c33342e626f65626c696e67656e2e64652e69626d2e
636f6d0003000000044d4249050000000005000101000000090
300006200000001510000004342436f6e6e6563746f722d706f6
f6c33342e626f65626c696e67656e2e64652e69626d2e636f6d2d
706f6f6c33342e626f65626c696e67656e2e64652e69626d2e636
f6d2d4e616d652d5365727665724753535f444345004300dc0e0
0001400000008000000016f7e007e00bc0b***

IDL (Interface Definition Language)



Die Idee ist:

Über ein (binäres) Server Object ist nicht bekannt
wie es implementiert ist
in welcher Sprache es implementiert ist

Nur die Schnittstellen des Server Objektes werden veröffentlicht
Methoden
Parameter

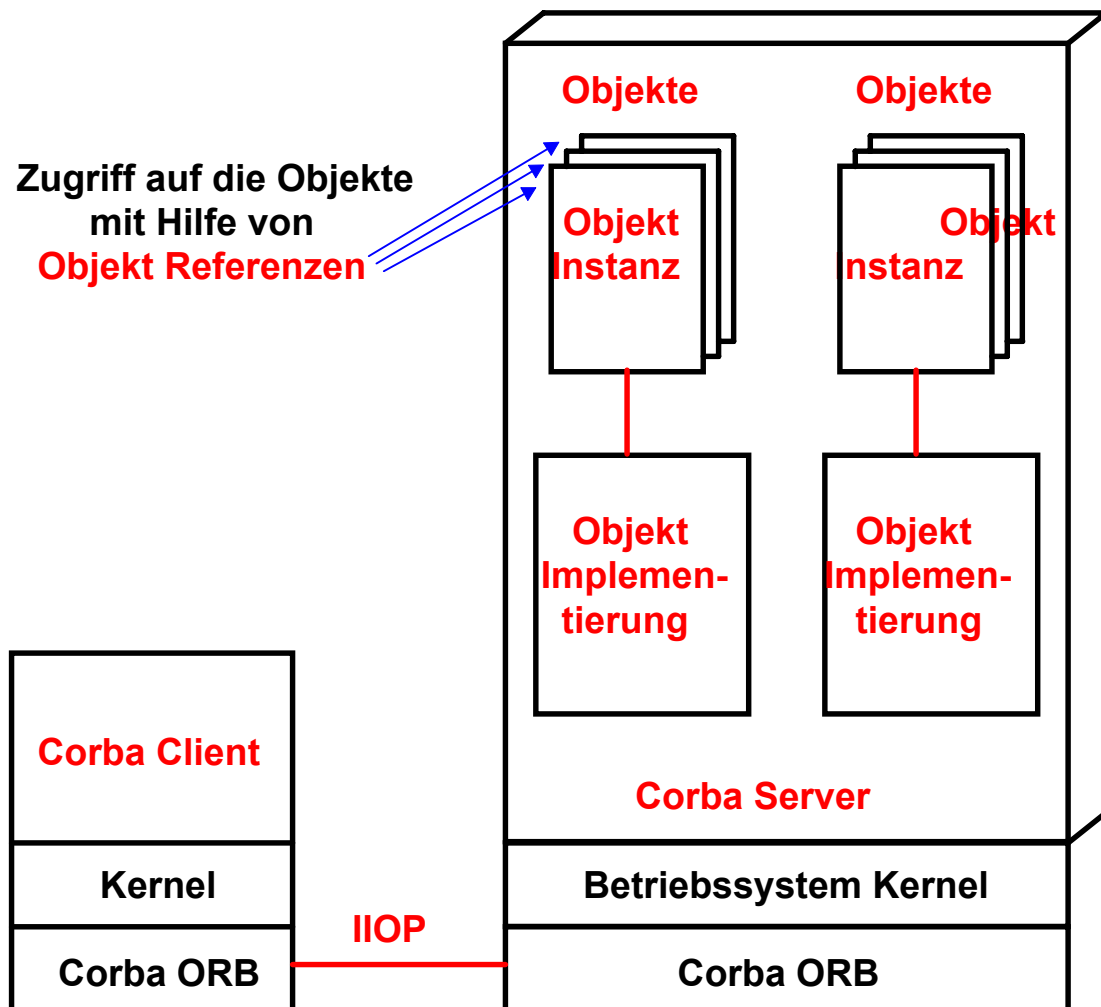
Um die Programmiersprachenunabhängigkeit zu erreichen, wird die
Schnittstellenbeschreibung von der binären (Maschinencode)
Implementierung der Klienten- und Serveranwendung getrennt

Die Beschreibung der Schnittstellen erfolgt in einer einheitlichen
Sprache: IDL . Damit entsteht eine Programmiersprachen-
unabhängige Class Description. Die „Language Mappings“ der
OMG definieren die Umsetzung

Wenn der Programmierer die Schnittstellenbeschreibung des
Server Objektes kennt, kann er sie benutzen um eine
Klientenanwendung zu schreiben

Die Compilierung der IDL Schnittstellenbeschreibung erzeugt auf
der Klienten- und Serverseite ORB spezifische Skeletons und Stubs

Der Skeleton und Stub Code muß Sprachen-spezifisch sein, um mit
der Klienten - und Serveranwendung in deren Sprache zurecht zu
kommen



Corba Begriffe

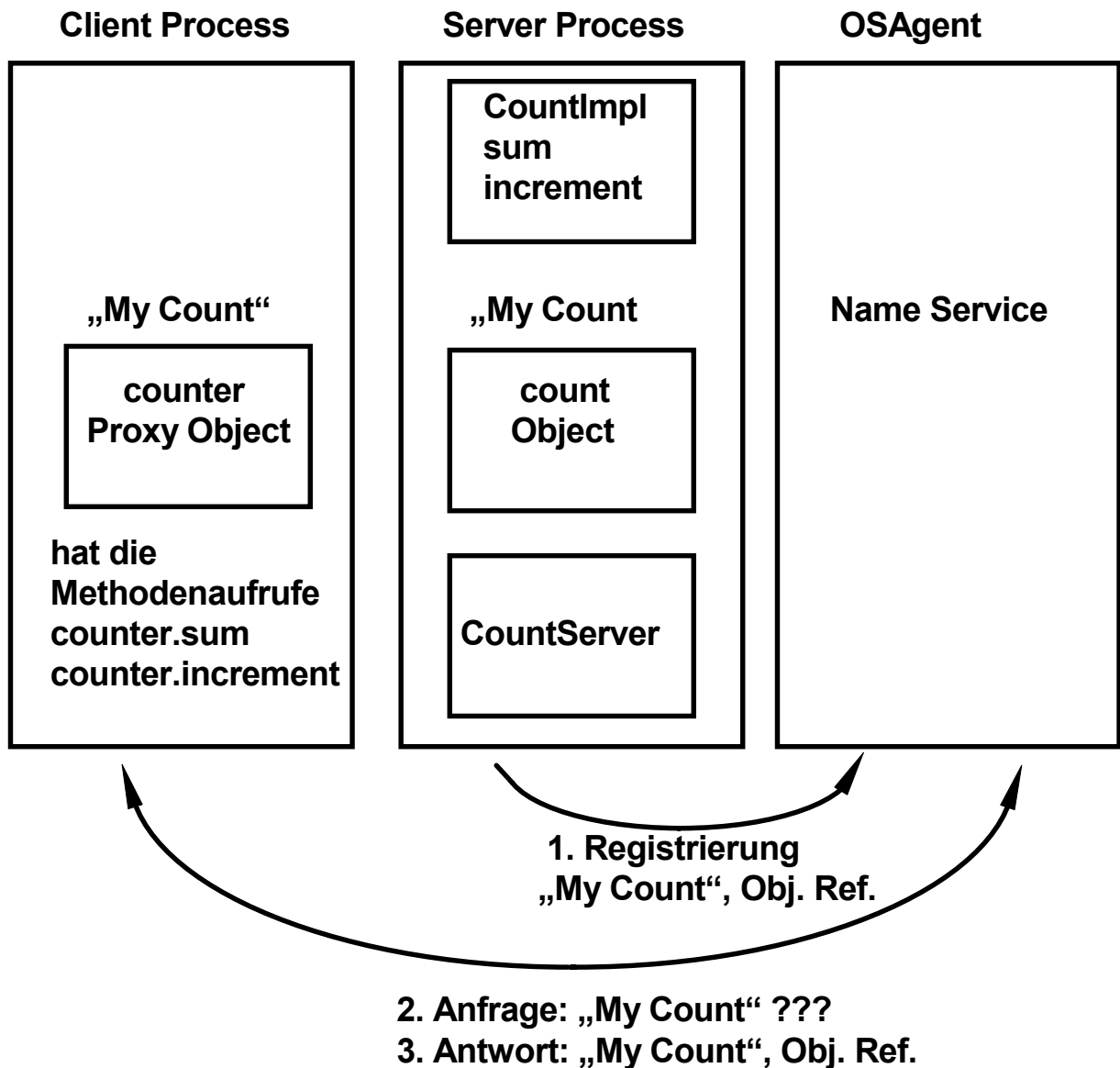
Objekt Referenz : Zeiger auf ein Objekt, das irgendwo im Netzwerk existiert

Corba Client : Programm, das eine Objekt aufruft

Objekt Implementierung : Programm Code, der das Verhalten des Objekts definiert

Objekt Instanz : Ausprägung eines bestimmten Objektes

Corba Server : Prozess, der eine oder mehrere Objekt Implementierungen bereitstellt



Der interne „state“ des Objektes „My Count“ wird durch 1 Variable (int sum) dargestellt.

Counter Beispiel

```
// Count.idl  
  
module Counter  
{  
    interface Count  
        { attribute long sum;  
          long increment();  
        };  
};
```

IDL Schnittstellen Definition der Count Schnittstelle

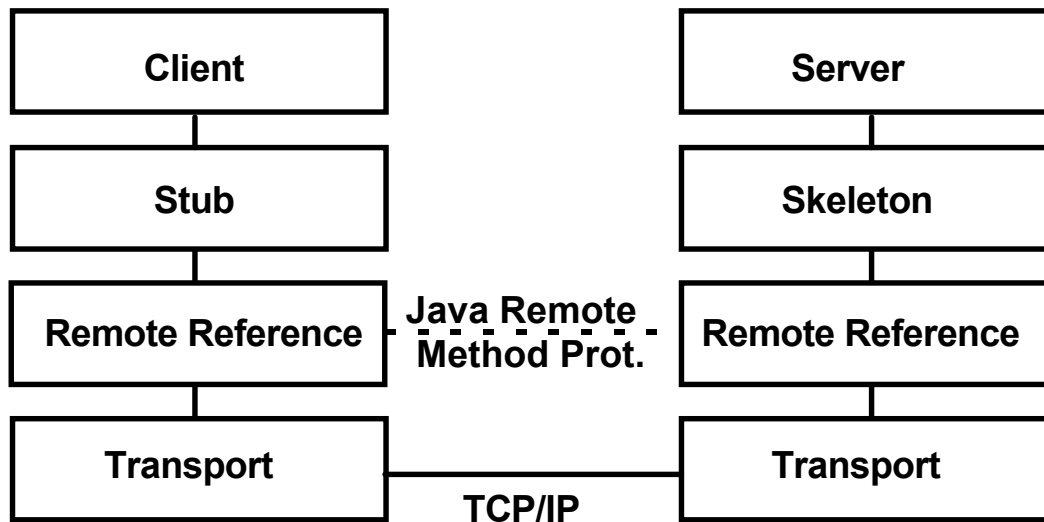
Der IDL Precompiler erzeugt aus der Schnittstellen Definition Java Programme für

**Client Stub
Server Skeleton,
weitere Hilfsprogramme**

und lädt die Schnittstellen Beschreibung in das Interface Repository.

Remote Method Invocation (RMI)

Aufruf von Java Programmen auf geographisch entfernten Rechnern



JRMP - Java Remote Method Protocol

Klient besorgt sich eine Handle für das entfernte Objekt, indem er RMI Registry aufruft (//URL/registered name).

Eine Referenz auf das entfernte Objekt wird zurückgegeben. Jetzt kann eine Methode des entfernten Objektes aufgerufen werden.

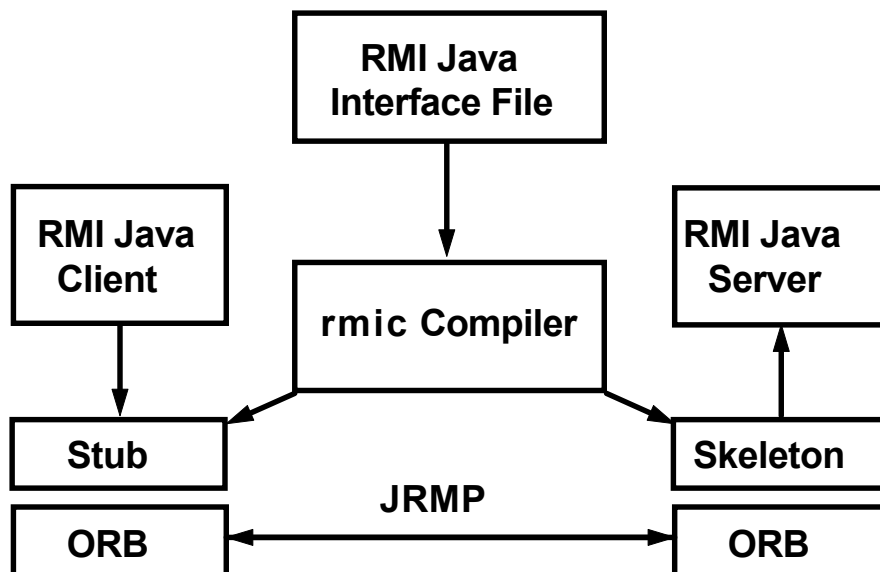
Dieser Aufruf erfolgt zum Stub, der das entfernte Objekt repräsentiert.

Der Stub verpackt die Argumente (marshaling) in einen Datenstrom, der über das Netzwerk geschickt wird.

Das Skeleton unmarshals die Argumente, ruft die Methode, marshals die Ergebnisswerte und schickt sie zurück.

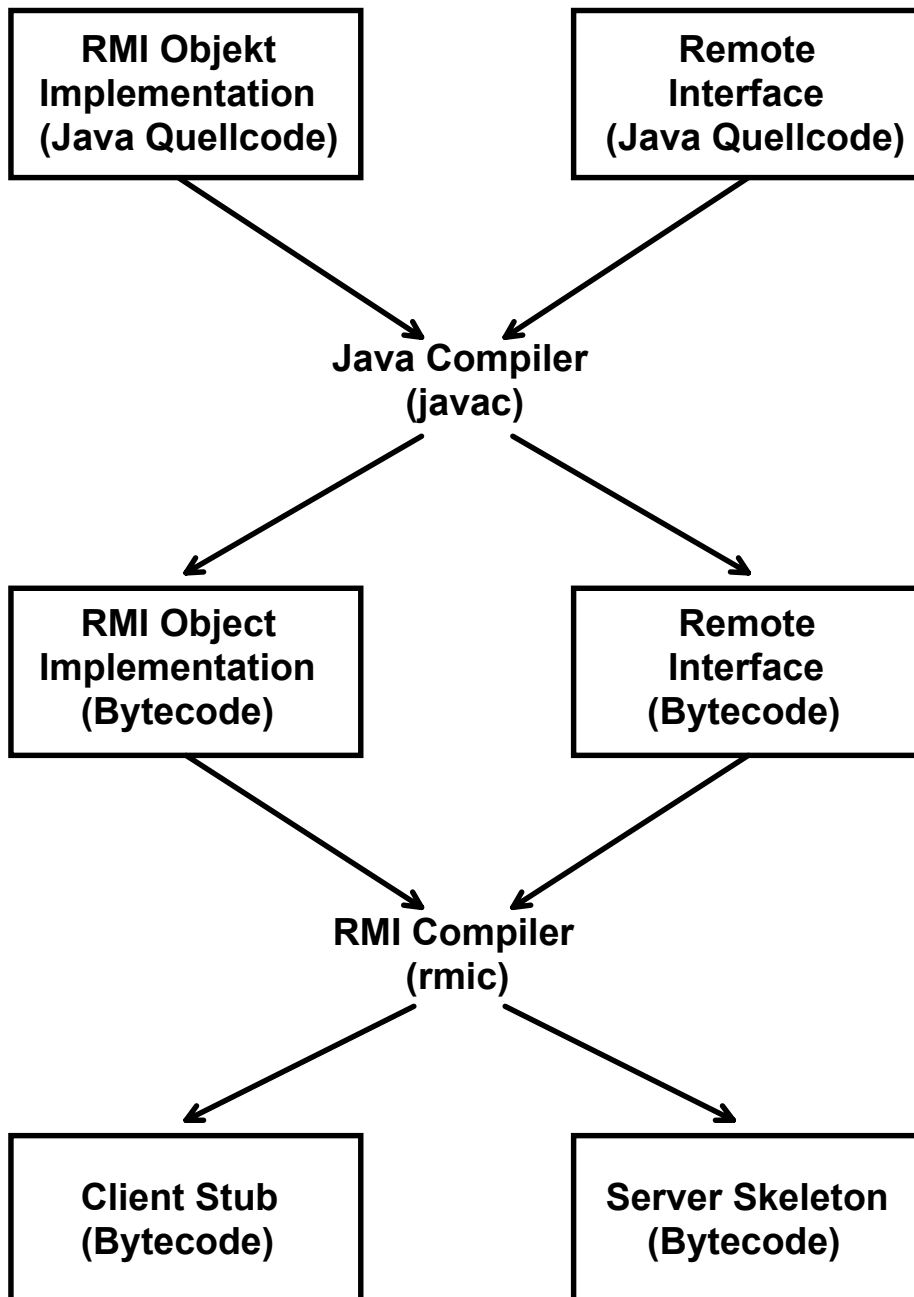
Der Stub unmarshals die Ergebnisswerte und übergibt sie an das Klientenprogramm.

Remote Class - Serializable Class



Erstellen einer RMI Remote Class

1. Interface definieren, mit der das Remote Object aufgerufen wird.
 > `extends interface java.rmi.Remote .`
2. Implementierung der Server Anwendung schreiben. Muss die Remote Interface implementieren. .
 > `extends java.rmi.server.UnicastRemoteObject`
3. Klassen kompilieren.
4. Mit dem Java RMI Compiler Client Stubs und Server Skeletons erstellen. > `rmic xyzServerImpl`
5. Klient implementieren und übersetzen.
6. Start Registry, Server starten, Klienten starten.



Entwicklungsprozess für RMI Objekte

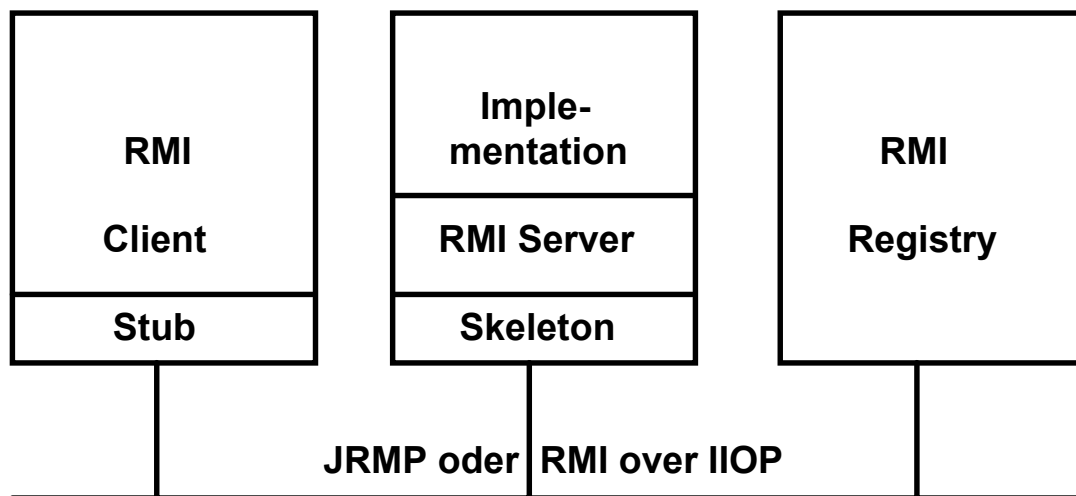
Die Remote Interface enthält die Namen aller Methodenaufrufe und die dazugehörigen Parameter. Beispiel für die Interface einer Methode Addition, die zwei Zahlen a und b addiert:

```
public interface Addition extends java.rmi.Remote {  
    public long add(long a, long b)  
    throws java.rmi.RemoteException;  
}
```

RMI benötigt wie Corba einen RMI Server, unter dem die RMI Implementation läuft.

Zu kodieren sind:

- Interface
- Client
- Implementation
- Server

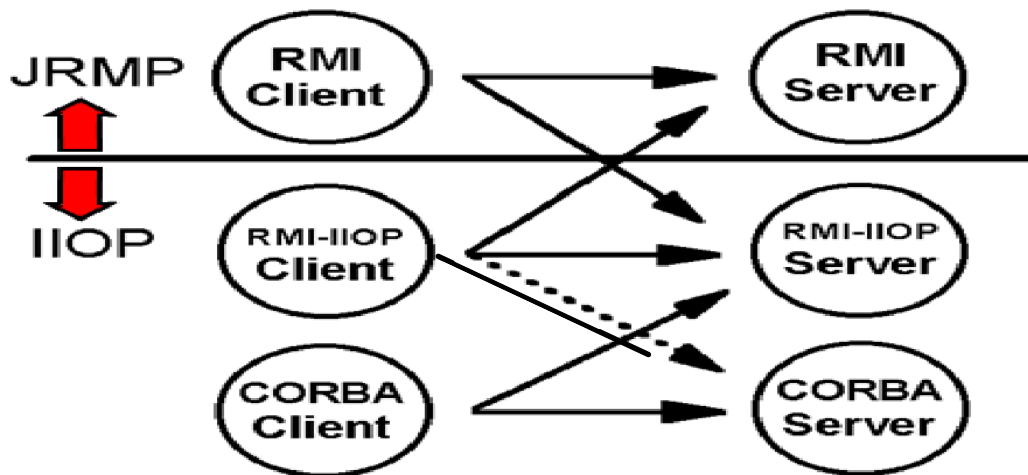


Drei verschiedene Prozesse, die auf dem gleichen Rechner oder auf entfernten Maschinen laufen können. RMI Registry ist ein einfacher RMI Namensdienst. Alternative JNDI

Code Beispiel: <http://developer.java.sun.com/developer/onlineTraining/rmi/RMI.html#IntroRMI>

Vergleich CORBA - RMI

- **Corba Anwendungen können in vielen unterschiedlichen Sprachen geschrieben werden, solange für diese Sprachen ein „interface definition language (IDL) mapping“ vorhanden ist. Dies schließt C/C++, Ada, Fortran, und Cobol ein; weitere Sprachen sind in Vorbereitung. RMI ist auf Java beschränkt.**
- **Mit der IDL ist die Interface von der Implementation sauber getrennt. Es können unterschiedliche Implementierungen unter Benutzung der gleichen Interface erstellt werden.**
- **RMI Anwendungen sind einfacher zu erstellen als Corba Anwendungen, weil die Notwendigkeit der IDL Definition entfällt-**
- **RMI ermöglicht serialisierbare Klassen. Code und Objekte können über das Netz übertragen werden (can be marshaled), solange der Empfänger über eine Java Virtuelle Maschine (JVM) verfügt. CORBA erlaubt keine Übertragung von Code oder Objekten; es können nur Datenstrukturen übertragen werden.**
- **Corba hat ein besseres Leistungsverhalten als RMI (keine Interpretation).**



RMI-over-IIOP

Corba verwendet das IIOP Protokoll für die Client-Server Kommunikation

RMI verwendet das JRMP Protokoll für die Client-Server Kommunikation

Der Java J2EE Standard und sein JDK unterstützt „RMI-over-IIOP“ als zusätzliche Alternative zu dem bisherigen „RMI-over-JRMP“. Dies bedeutet, es wird IIOP an Stelle von JRMP als Übertragungsprotokoll eingesetzt.

Es wird der `rmic-iiop` Compiler an Stelle des bisherigen `rmic` Compilers eingesetzt, um die RMI Stubs und Skeletons zu erzeugen. Davon abgesehen ändert sich wenig an der Erzeugung und dem Betrieb von RMI - Java Anwendungen. Einige Hersteller, z.B. IBM, wollen deshalb RMI-over-JRMP ganz durch RMI-over-IIOP ablösen.

Soll eine RMI Komponente (in Java) mit einer Corba Komponente (z.B. in C++) kommunizieren, so ist es erforderlich, aus der Java Interface Definition mit Hilfe des `rmic-idl` Compilers eine IDL File zu erstellen. Letztere erzeugt dann mit Hilfe des normalen IDL Compilers Stubs und Skeletons.



Microsoft DCOM

Das DCOM Objektmodell ist Microsoft's proprietäre Alternative zu CORBA.

Verwendet (leicht modifizierten) DCE RPC.

Integriert (wie Corba) Komponenten unterschiedlicher Hersteller in binärer Form, die in unterschiedlichen Sprachen geschrieben sein können. Verwendet hierzu eine IDL.

Keine Vererbung.

Verfügbar unter OS/390 und den meisten Unix Dialekten (incl. LINUX), jedoch bisher kaum Einsatz außerhalb der Windows Betriebssystemfamilie.

Vergleich CORBA - DCOM

Beide Technologien werden vermutlich für längere Zeit in Wettbewerb miteinander stehen.

In reinrassigen Microsoft Umgebungen (z.B. mittelständische Unternehmen) hat DCOM Vorteile.

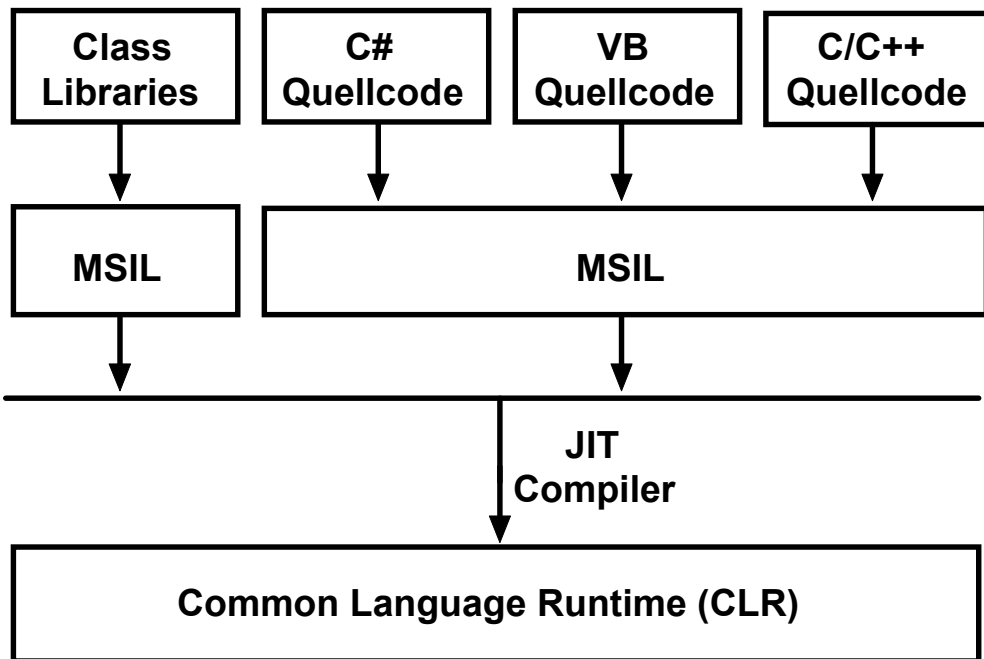
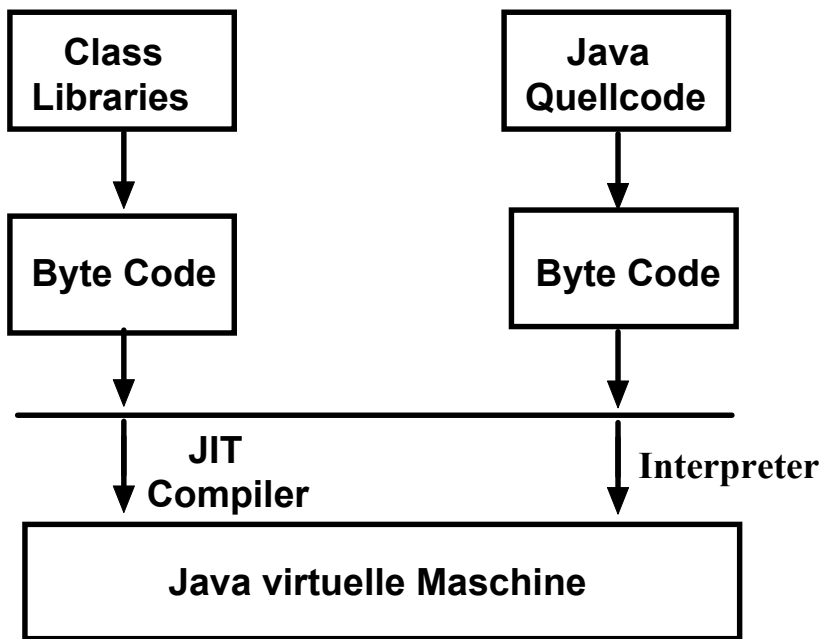
In heterogenen Umgebungen ist DCOM bisher wenig vertreten.

CORBA ist besser geeignet, bestehende Altanwendungen einzubinden.

Es ist denkbar, auf weitverbreitete Echtzeitbetriebssysteme wie VxWorks oder Psos eine CORBA basierte Anwendung aufzusetzen.

Java Browser Sicherheit (Sandbox) ist besser als DCOM (vertrauensbasierte Verfahren).

Gartner: DCOM wird 2002 den CORBA Stand von 1998 erreichen



Microsoft Common Language Runtime (CLR)

DCOM Leistungsverhalten

**Beispiel: Pentium 120 Mhz, NT 4.0 Betriebssystem,
50 Bytes an Parametern zwischen 2 Prozessen
transportieren**

- 1. Transport zwischen 2 Prozessen auf dem gleichen
Rechner ohne DCOM
3 Millionen Aufrufe/s**
- 2. Transport zwischen 2 Prozessen auf dem gleichen
Rechner unter Verwendung von DCOM und dem MT LPC
2000 Aufrufe/s**
- 3. Transport mit DCOM zwischen 2 Prozessen auf
unterschiedlichen Rechnern
500 Aufrufe/s**

Web Services

Beispiel Yahoo Portal

Die meisten Dienste kommen in Wirklichkeit von anderen Web Sites: Reisen, Wetter, Landkarten, oder Web Suche mit Hilfe von Google. Diese Dienste sind in das Yahoo Portal als eigene Dienste integriert.

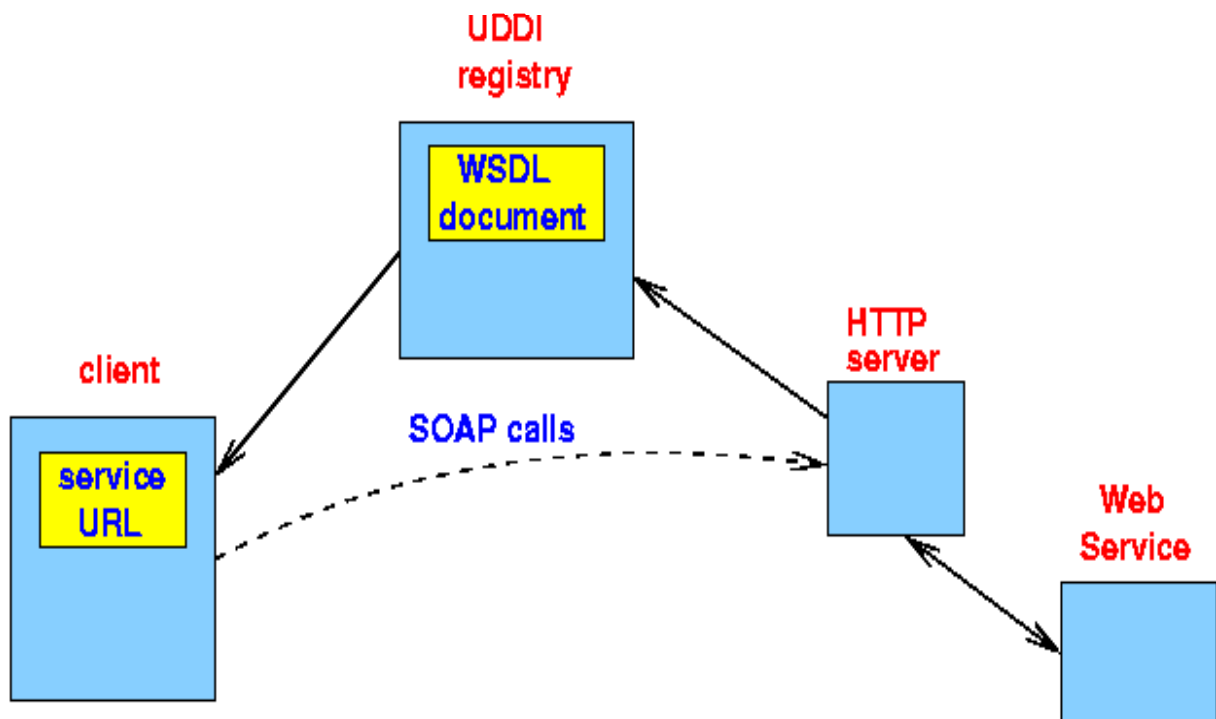
- **SOAP** Simple Object Access Protocol
- **WSDL** Web Service Description Language
- **UDDI** Universal Description, Discovery, and Integration

SOAP ist ein Remote Procedure Call (RPC) Protokoll. Es benutzt Standard Internet Protokolle für den Transport - entweder HTTP für synchrone Aufrufe oder SMTP für asynchrone Aufrufe. XML beschreibt das Format der übertragenen Daten. SOAP ist unabhängig von der verwendeten Programmiersprache, dem Objekt Modell und dem jeweiligen Betriebssystem.

WSDL beschreibt die Schnittstellendefinitionen eines Web Service. Beschreibt Formate der Anforderungs- und Antwort-Nachrichtenströme, mit denen Funktionsaufrufe an andere Programm-Module abgesetzt werden.

UDDI ist ein Dienstekatalog. Stellt ein Verzeichnis von Adress- und Produktdaten sowie Anwendungsschnittstellen der verschiedenen Web Service Anbieter dar.

IBM - Microsoft Kooperation
Firewall - HTTP
Reifegrad: Sicherheit, Transaktionsdienste.



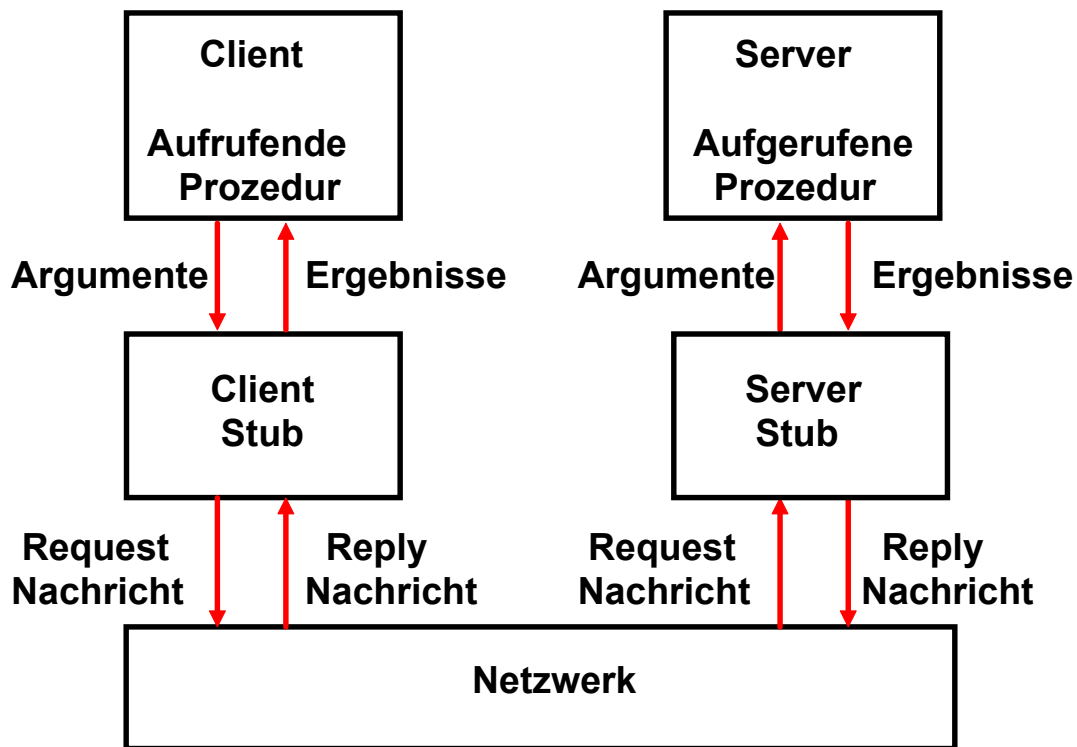
Web Services Struktur

Eigenschaften:

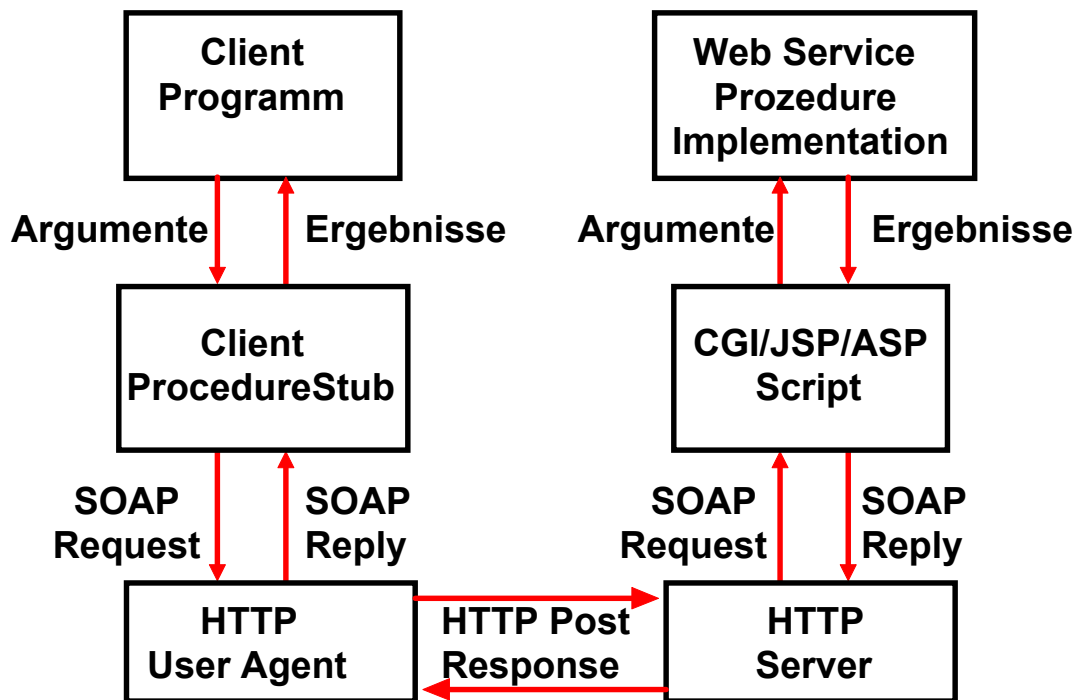
- Quick and dirty Implementierung von unkomplizierten Aufgaben
- Remote Procedure Call über Port 80
- WSDL an Stelle von IDL - automatische Erstellung, z.B. IBM's Web Services ToolKit (WSTK)
- Java → WSDL → Java - identische Ergebnisse ?
- keine Objektorientierung
- skaliert schlecht

Was fehlt:

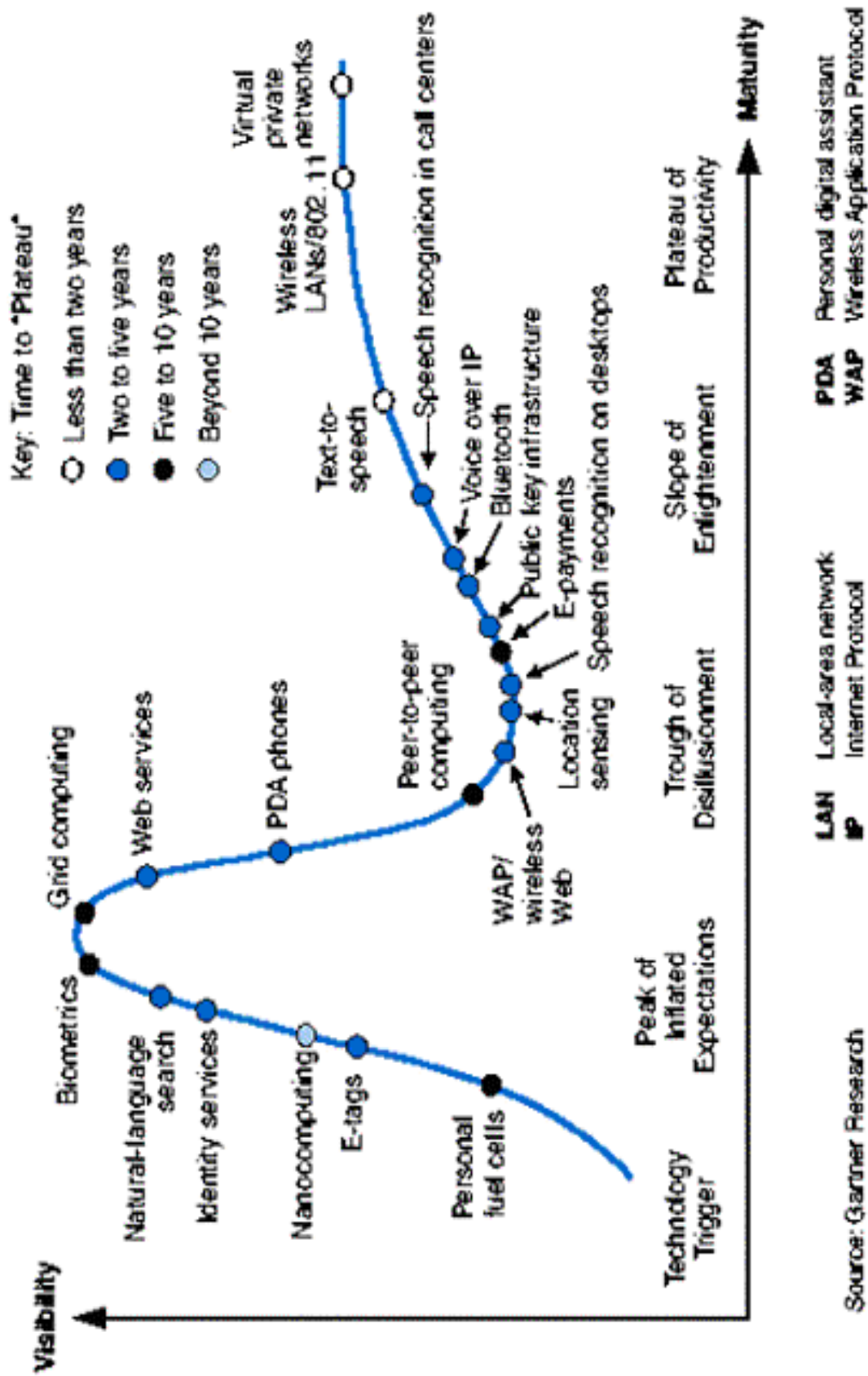
- Sicherheit - HTTPS kann benutzt werden, ist aber unabhängig vom Web Service Mechanismus
- Transactionssteuerung, Flußsteuerung



Remote Procedure Call



Web Services Remote Procedure Call



e-Business

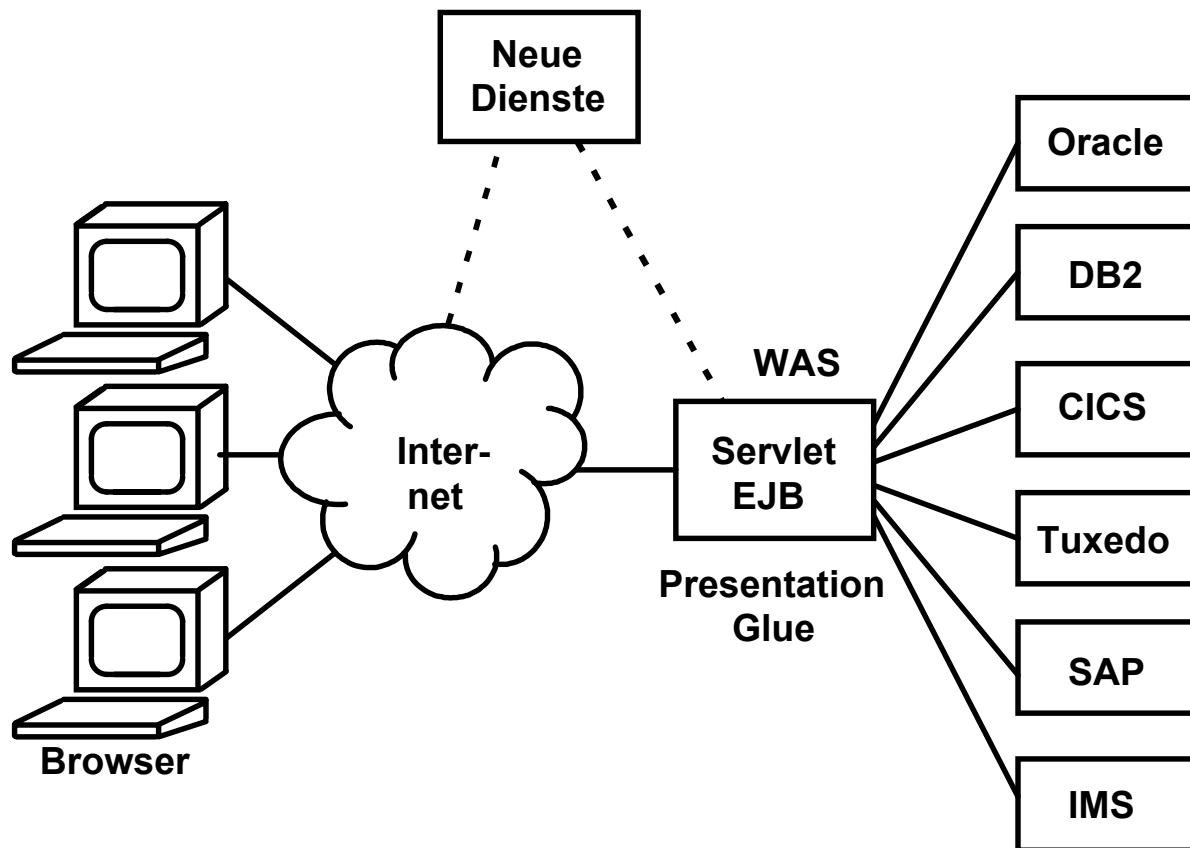
Erweiterung der Internet Technologien in Richtung IT Integration

- **zwischen Unternehmen (business to business, B2B)**
- **zwischen Unternehmen und Endverbraucher (business to customer, B2C)**

Besteht aus den Teilen:

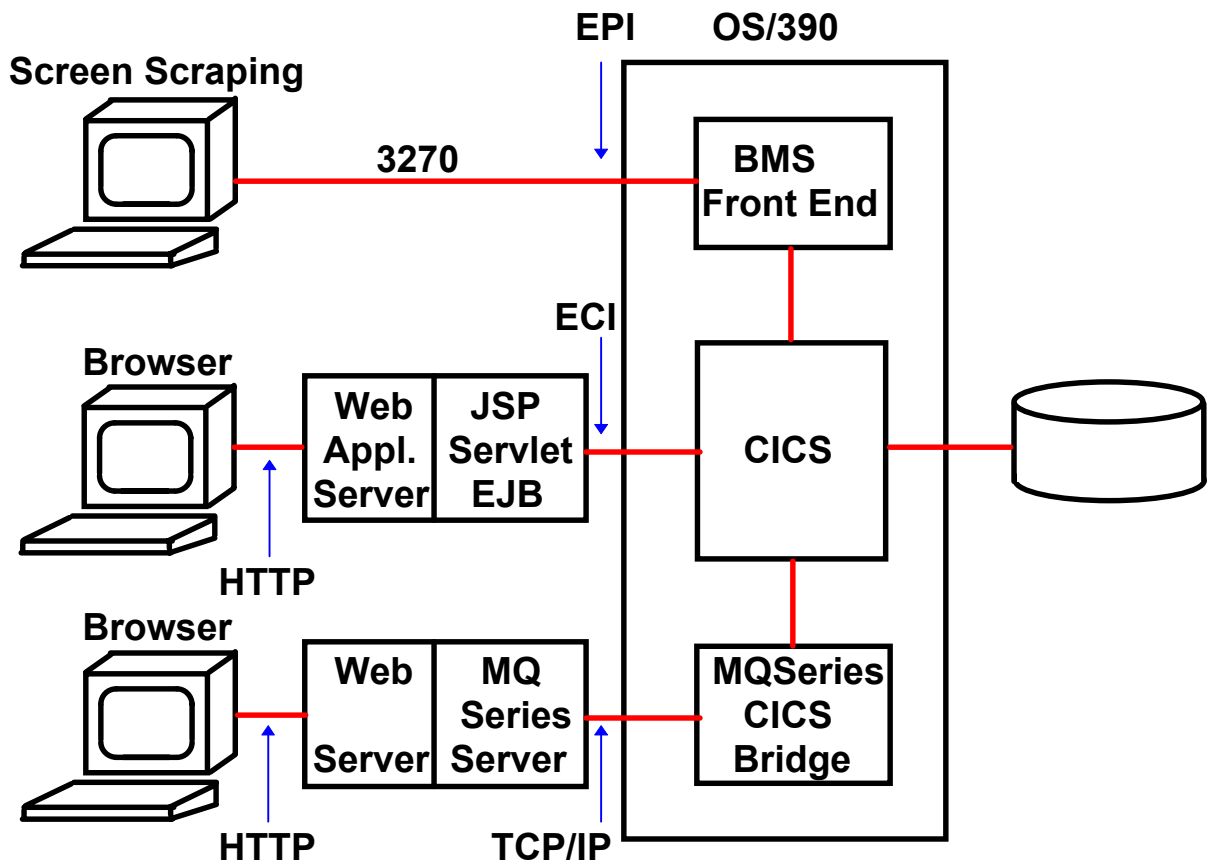
- **Supply Chain Management (SCM)**
- **Customer Relation Management (CRM)**
- **Electronic Commerce**
- **Business Intelligence**

Das Erstellen von e-Business Anwendungen erfordert spezielle Werkzeuge.



Client/Server Aufgabenstellung

- Browser orientierter Web Zugang
- Datenhaltung in existierenden Datenbanken
- Dominierender Anteil der Business Logik in existierenden Transaktionsprogrammen und/oder Stored Procedures
- Neue Software (z.B. EJBs) stellen Querverbindungen zwischen existierenden Komponenten her (Glue)
- System Management - TCO

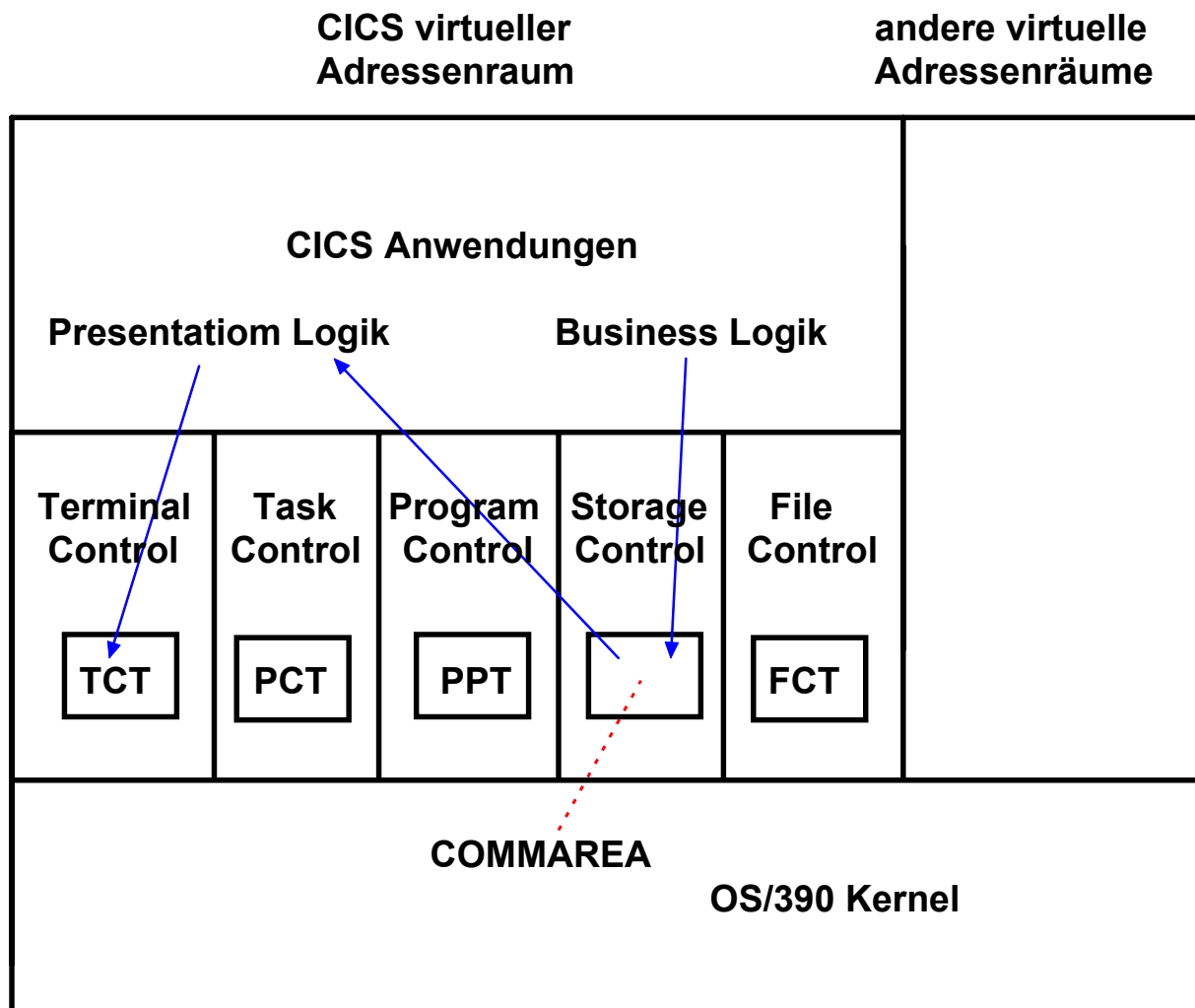


CICS Klienten Anbindung

EPI Die BMS Maps werden weiter verwendet. Keine Änderung der Information, die auf dem Bildschirm wiedergegeben wird. Die Darstellung der Information kann geändert werden.

ECI Die Presentation Service Komponente von CICS (BMS) wird nicht genutzt. Direkter Zugriff auf COMMAREA.

MQSeries Asynchrone Übertragung durch Message oriented Middleware

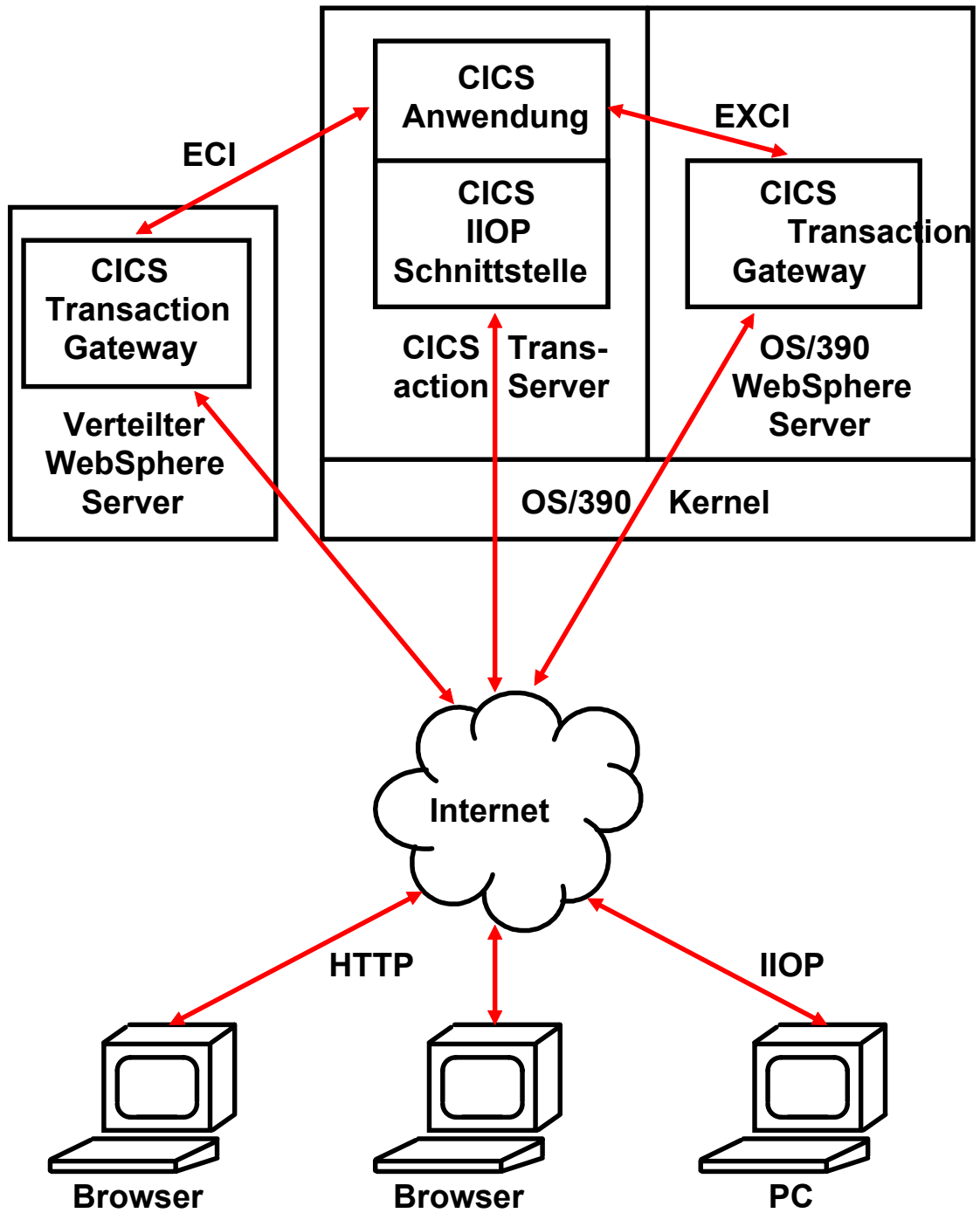


COMMAREA

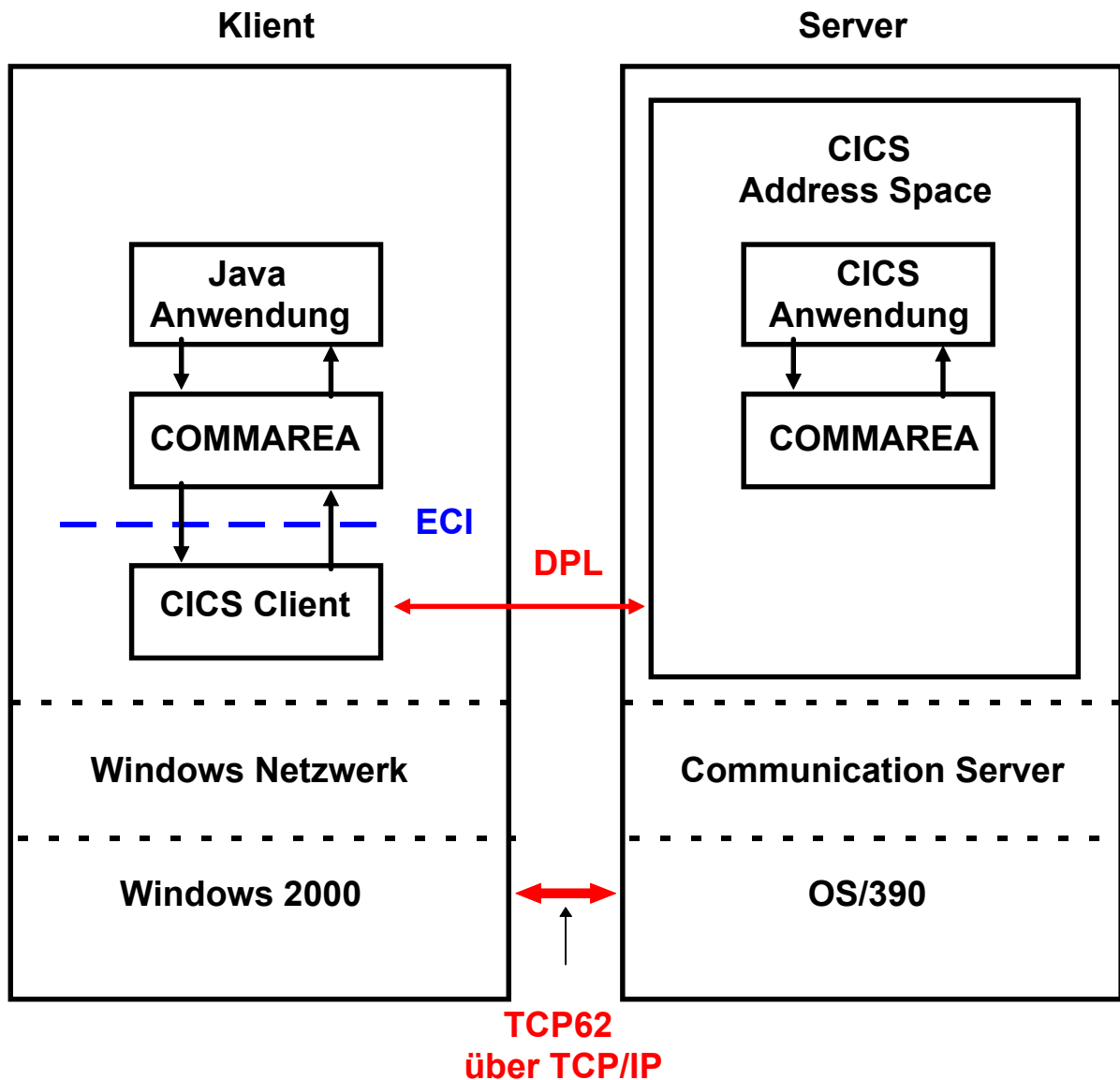
Communication Area

COMMAREA kann benutzt werden um Ein/Ausgabedaten zu übergeben:

- An ein CICS Anwendungsprogramm im gleichen Adressenraum
- An ein CICS Anwendungsprogramm in einem anderen Adressenraum auf dem gleichen physikalischen Rechner,
- An ein CICS Anwendungsprogramm auf einem getrennten physikalischen Rechner unter Benutzung von EXEC CICS LINK (...)
- An ein nicht unter CICS laufendes Programm, z.B. ein GUI Prozess unter Verwendung der EPI Schnittstelle



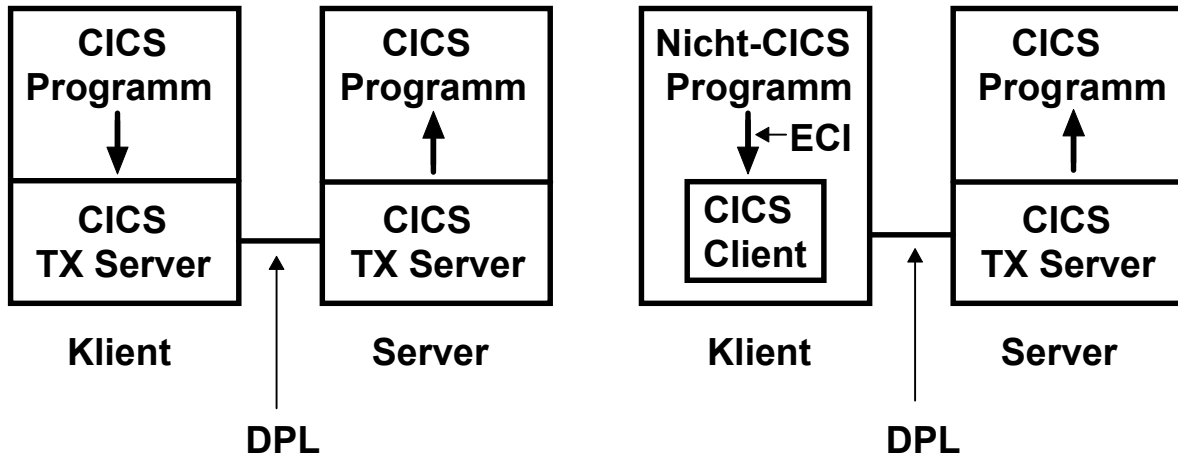
Alternative CICS Zugriffe über das Internet



Ein CICS Client ist eine echte CICS Anwendung, die mit anderen CICS Anwendungen über Distributed Program Link (DPL, einem RPC ähnlichen Mechanismus) verkehrt.

Eine Java Client Anwendung kann über die ECI Schnittstelle auf den CICS Client zugreifen. Dies ermöglicht einen direkten COMMAREA Datenaustausch zwischen Klienten und Server. Die Beschränkungen des BMS/3270 Datenprotokolls (z.B. keine Scroll Bar) werden damit umgangen.

Ein CICS Anwendungsprogramm ruft die Dienste des CICS Transaktionsservers mit Hilfe von EXEC CICS Befehlen auf.



CICS Distributed Program Link (DPL)

DPL ist ein Remote Procedure Call Mechanismus, mit dem ein CICS Programm ein anderes CICS Programm aufrufen kann-

```
EXEC CICS LINK PROGRAM(name) COMMAREA(data-area)
```

ECI Call in C/C++

Die ECI (External Call Interface) ist eine Programmierschnittstelle, mit der eine Nicht-CICS Anwendung eine CICS Anwendung auf einem getrennten Rechner aufrufen kann.

```

ECI_PARMS          EciBlock;
cics_ushort_t      Response;
.....
.....
Response = CICS_ExternalCall (&EciBlock);

```

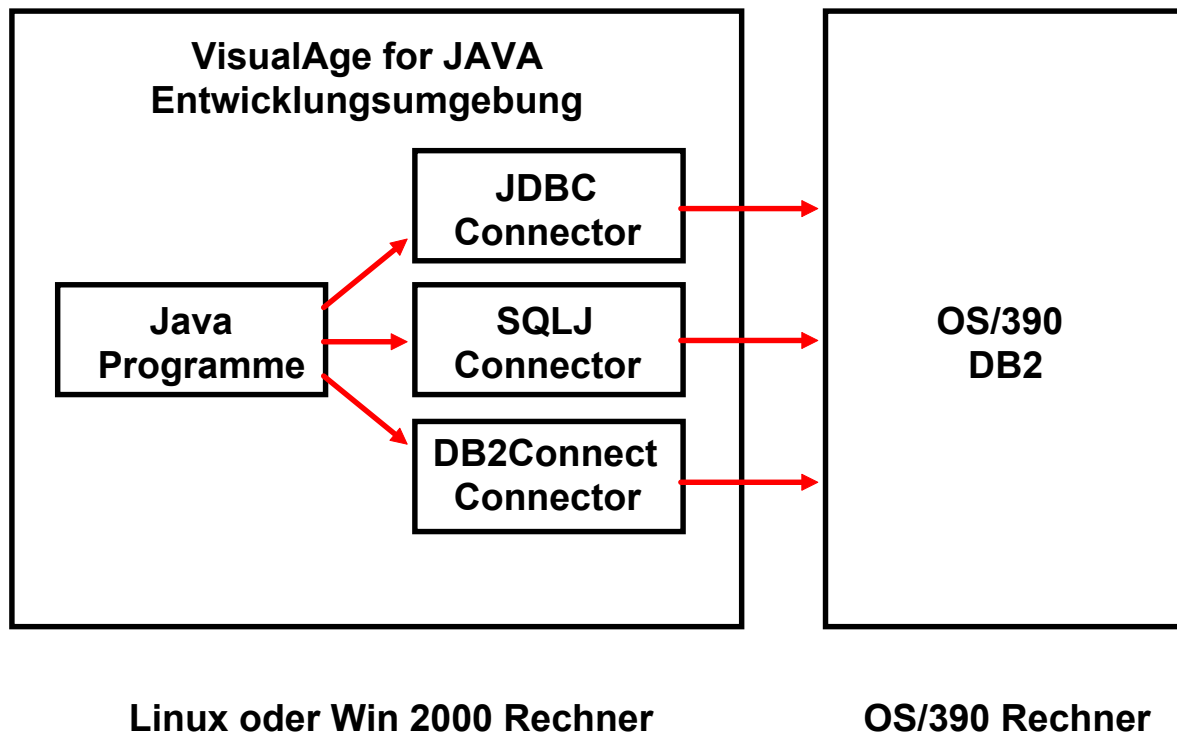

Connectors

Connectoren sind Java Beans, welche eine Schnittstelle zu existierenden Legacy Systemen bilden. Beispielsweise sind folgende Connectoren für die IBM WebSphere verfügbar:

- o JDBC
- o DB2
- o Oracle
- o Adabas
- o CICS
- o IMS
- o MQSeries
- o SAP R/3
- o Lotus Domino

Bei den Implementierungen von Web Commerce Lösungen spricht man vom Frontend, welches typischerweise mit einem Web Application Server realisiert wird, und einem Backend (Beispiele Auftragseingang, Finanzbuchhaltung), wofür vorhandene Legacy Systeme eingesetzt werden.

In vielen Fällen werden 20 % des Projektaufwandes für die Neuentwicklung des Frontends und 80% für dessen Integration in das vorhandene Backend aufgewendet.

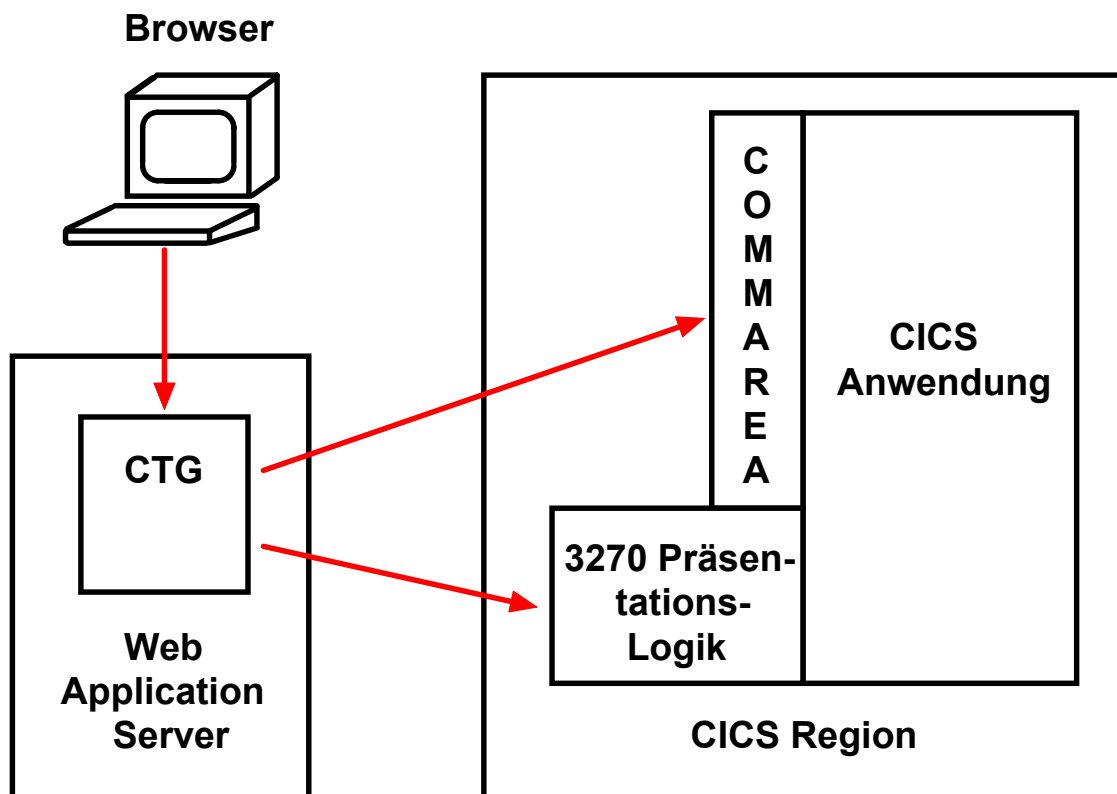


Unterschiedliche Connector Arten

JDBC ist von ODBC abgeleitet, implementiert dynamische Datenbankzugriffe

SQLJ implementiert statische Datenbankzugriffe

DB2Connect ist ein für die DB2 API optimierter Connector



CICS Transaction Gateway

Das CICS Transaction Gateway (CTG) ist eine von mehreren verfügbaren CICS Client Implementierungen. Es ermöglicht die Reduzierung der Klienten Funktionalität auf einen regulären Web Browser.

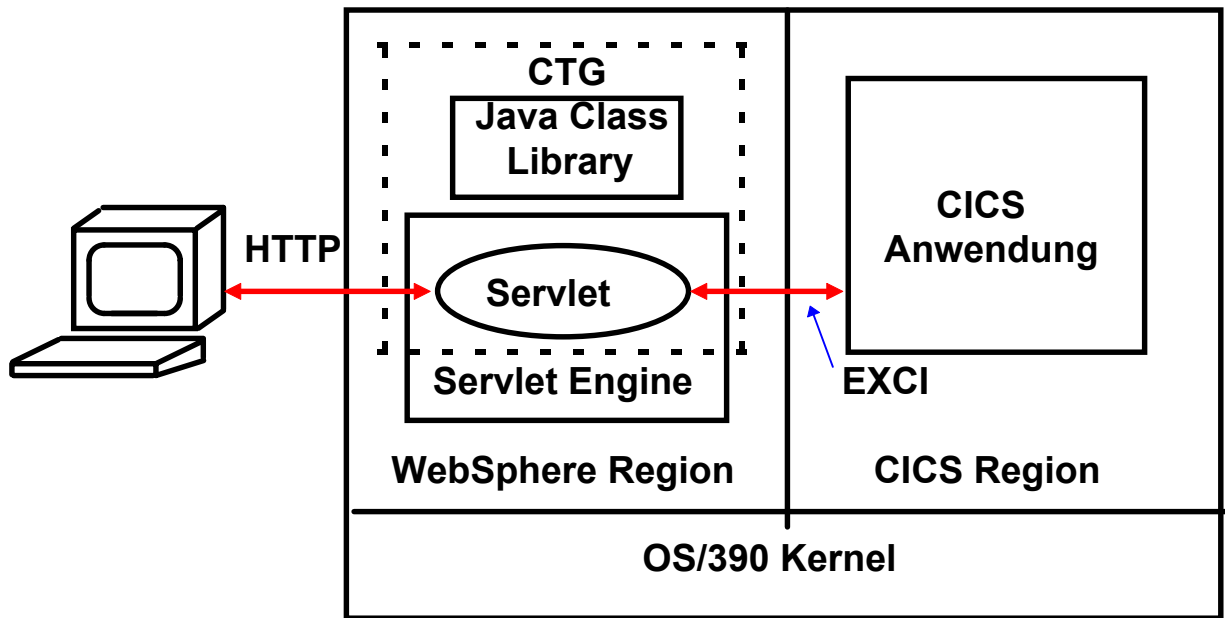
Als 2-Tier oder 3-Tier Konfiguration verfügbar

Vorteile der 2-Tier Konfiguration:

- Kein SNA (TCP/62 oder TN3270))
- Keine Sicherheitsprobleme zwischen Tier 2 und Tier 3
- Administration/System Management

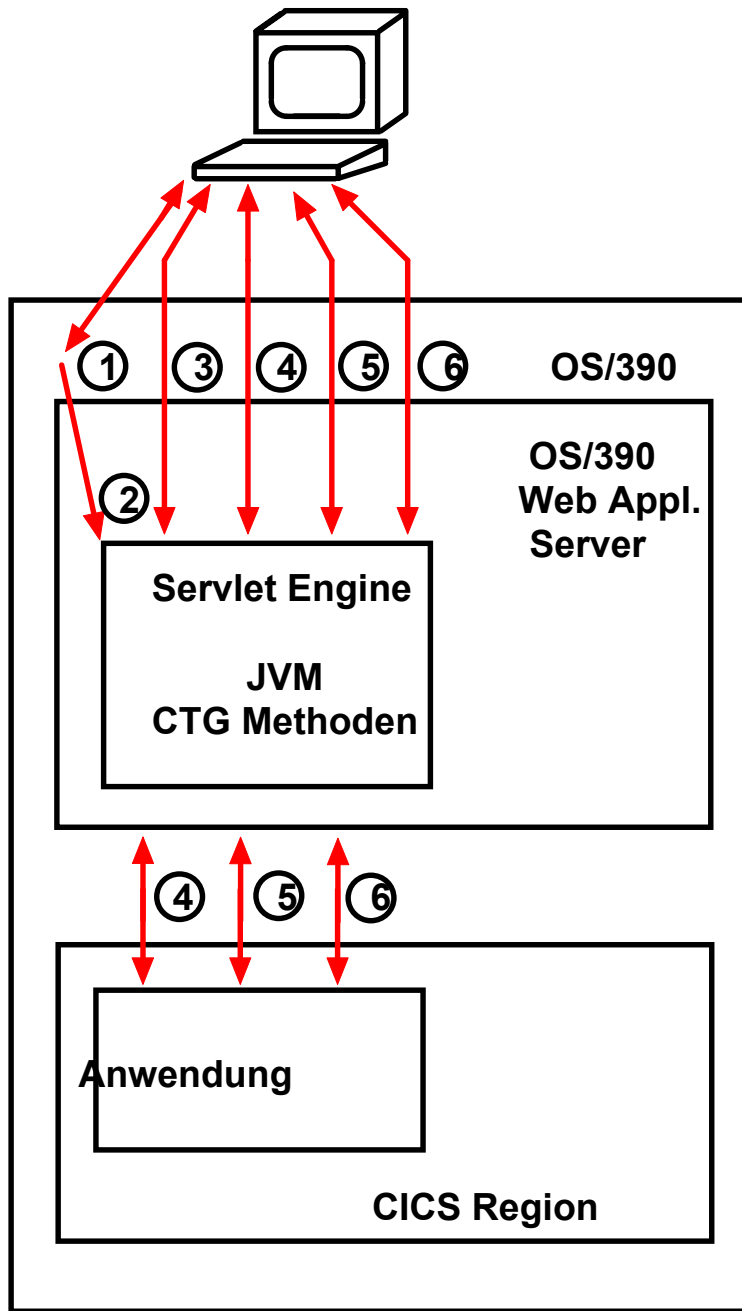
Nachteile der 2-Tier Konfiguration:

- 3-Tier ist quick and dirty
- Performance/Kosten
- Produktionsumgebung anders als Entwicklungsumgebung



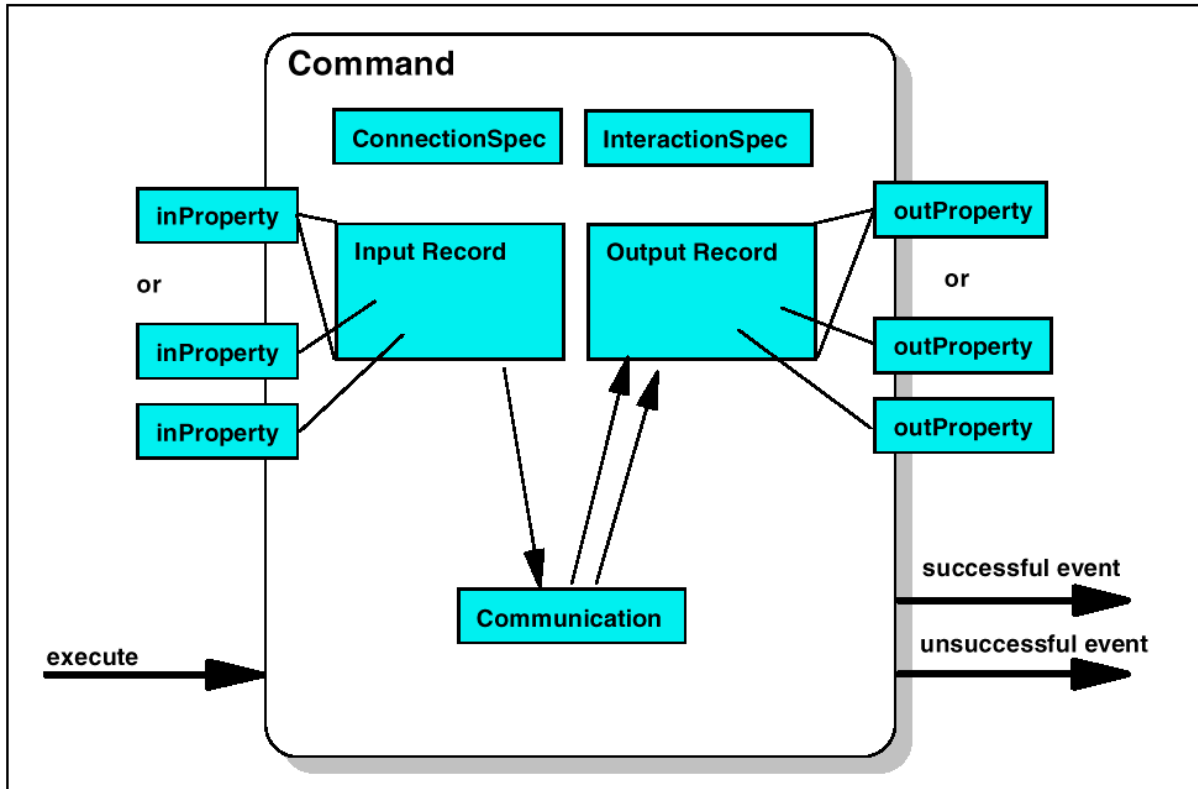
CICS Transaction Gateway Servlet Architecture

Diese Implementierung ist unter
<http://jedi.informatik.uni-leipzig.de> verfügbar



CICS Transaction Gateway

Common Connector Framework



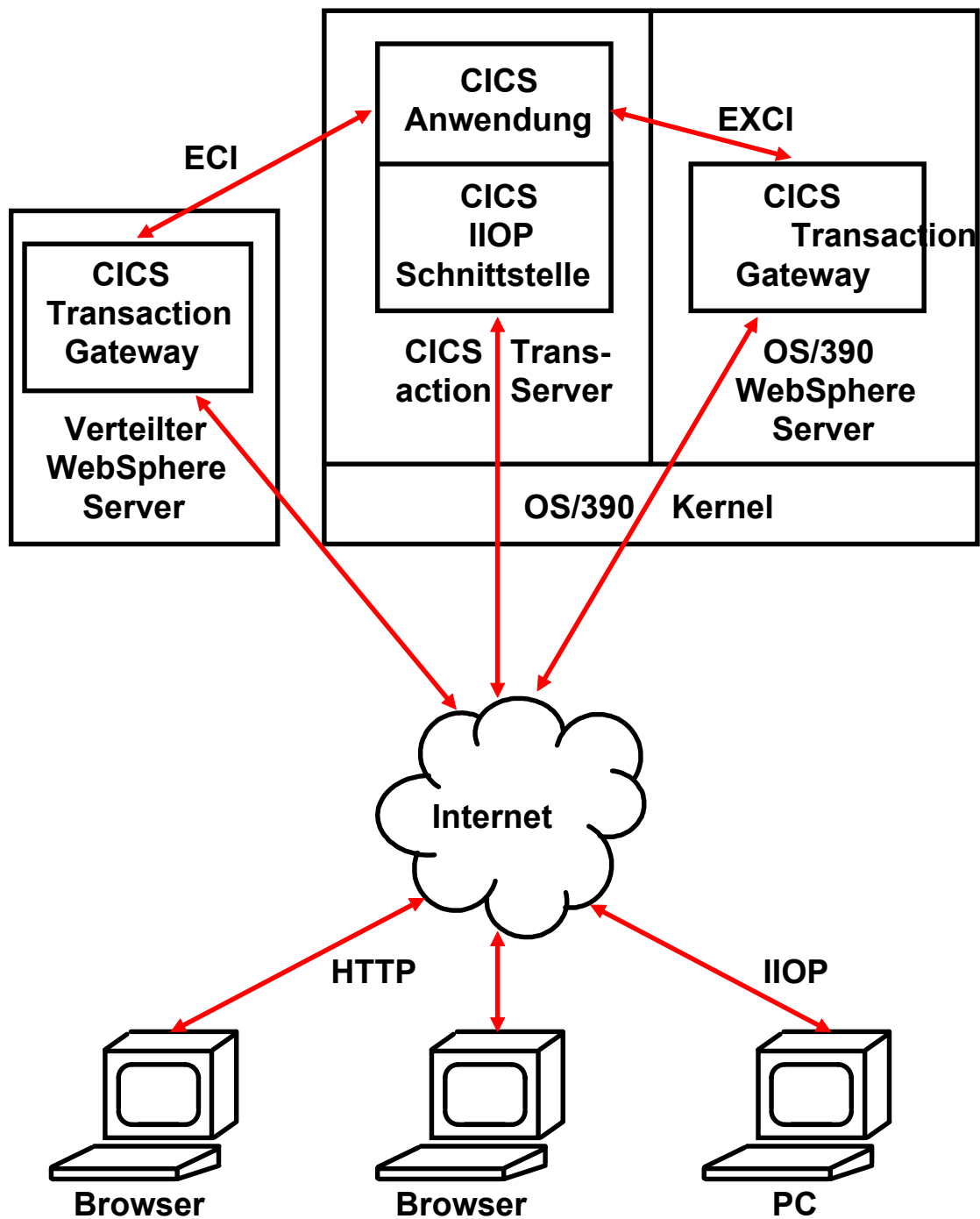
Aufruf des Common Connector Framework mit Hilfe des CCF Commands . Drei Schnittstellen: ConnectionSpec, InteractionSpec, und Communication, sowie Input und Output Records für die Datenübertragung.

ConnectionSpec spezifiziert eine Verbindung Attribute, wie Host Name, Port Nr., und Timeout Werte. Weitere Attribute können Connector spezifisch sein; beispielsweise ein Channel Identifier für einen MQSeries Connector.

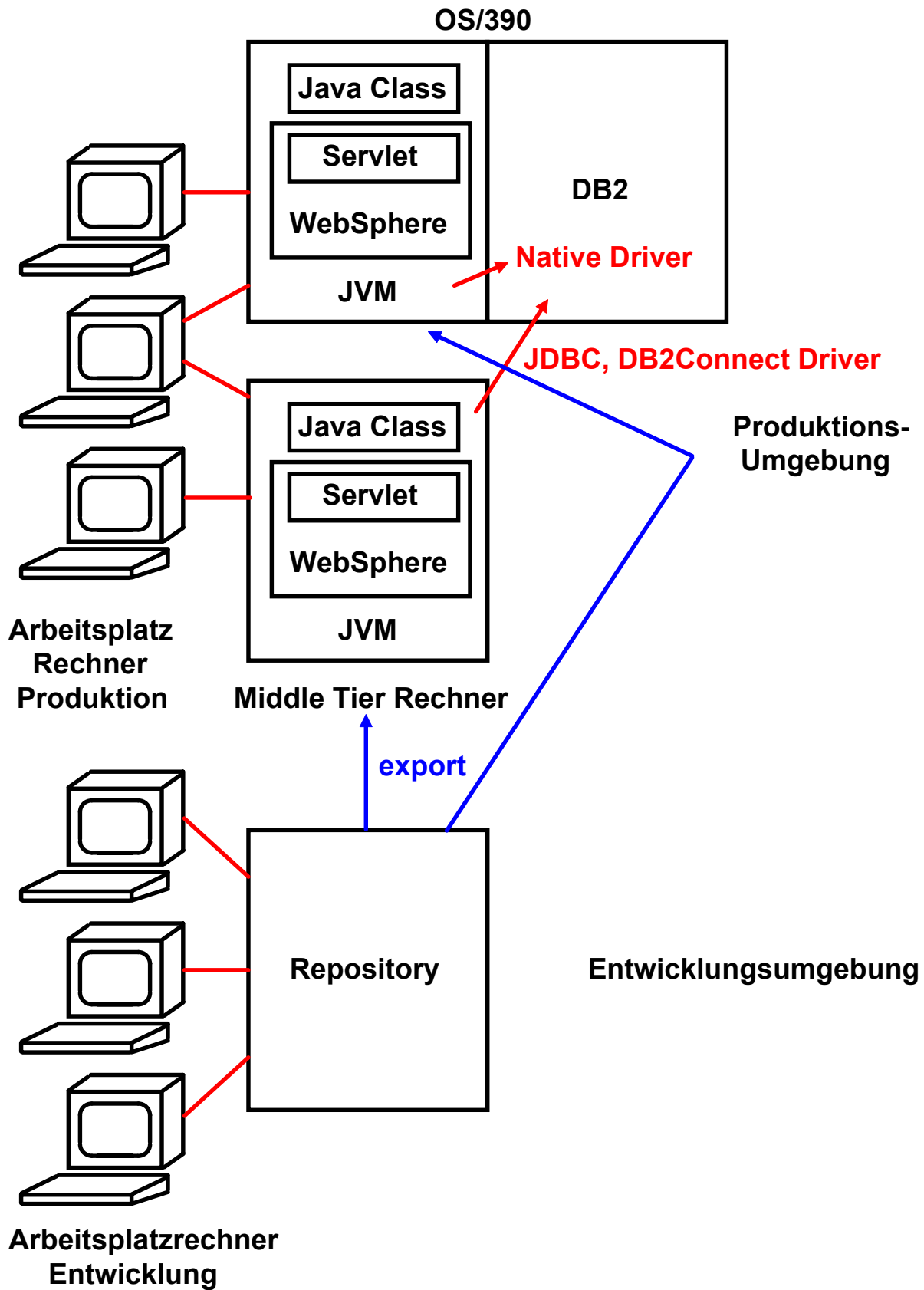
InteractionSpec enthält Angaben darüber was geschehen soll. Beispiele sind der Programmname oder Interaktionsmodus.

Eine Implementierung der Communication ruft deren Execute Methode auf. Die Execute Methode verwendet drei Argumente: InteractionSpec, Input Record und Output Record.

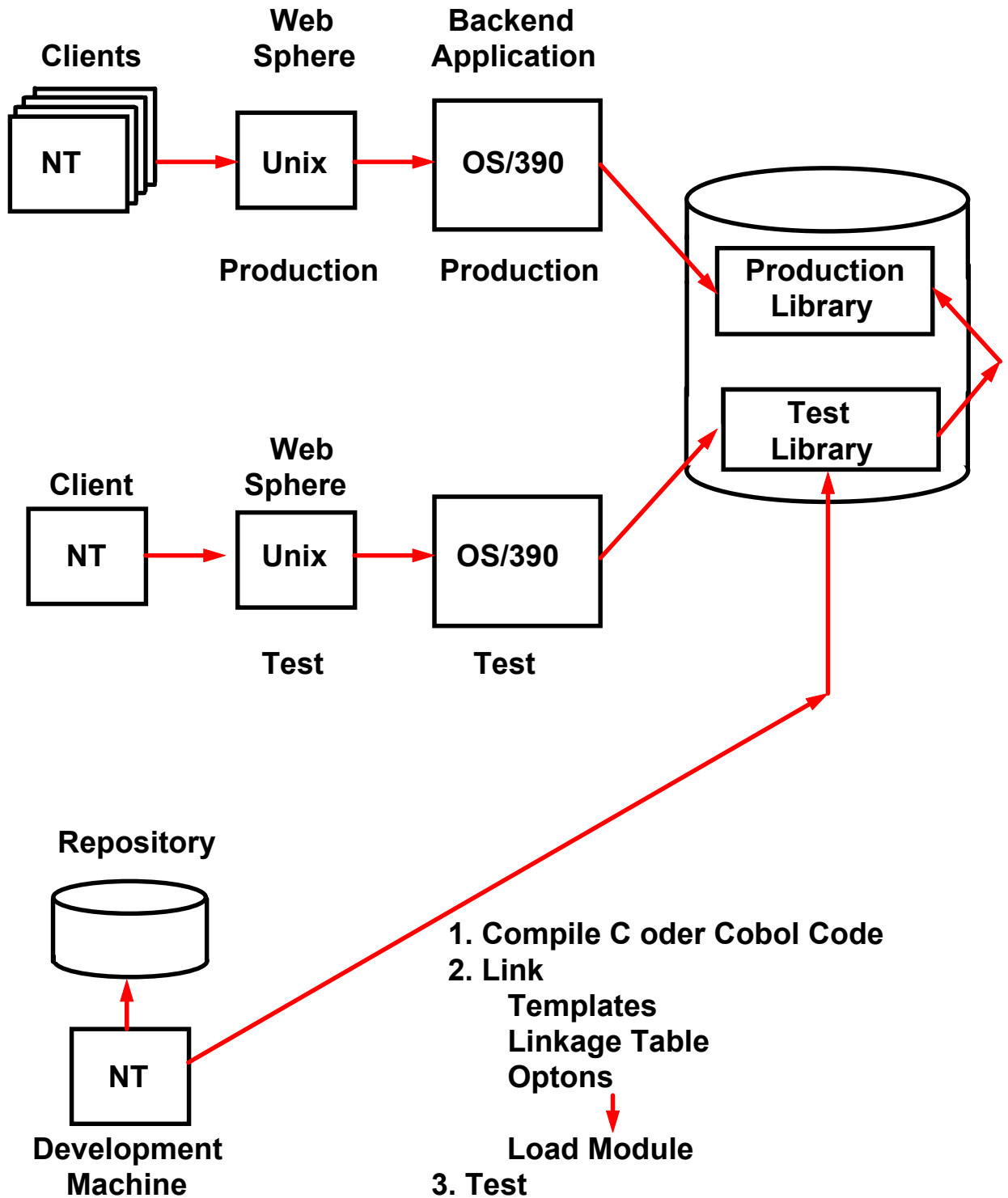
Input/Output Records repräsentieren die übertragenen Daten.



Alternative CICS Zugriffe über das Internet



Trennung zwischen Entwicklung und Produktion



Translating Development Code into Production Code

Client/Server Praktikum

WS 2003/2004

Prof. Dr.-Ing. Wilhelm G. Spruth

Es werden die folgenden Aufgaben bearbeitet:

- 1. LDAP Directory Service**
- 2. CORBA**
- 3. CORBA Namensdienst**
- 4. Java Servlet**
- 5. Programmierung unter OS/390 mit ISPF**
- 6. CICS BMS Transaktion - Zugriff auf eine DB2 Datenbank
(zählt doppelt)**

Für Freiwillige besteht die Möglichkeit zu weiteren Experimenten:

- 1. OS/390 Java Data Base Connectivity mit Eclipse für Java**
- 2. Java Client Zugriff mit MQSeries oder CTG auf CICS COMMAREA**

**Betreuung durch Frau Denkinger,
oliver.avieny@student..uni-tuebingen.de**

Die Vorlesung Client/Server Systeme ist als Wahlfach innerhalb der Technischen Informatik zugelassen, und kann mit 2 Stunden in den Prüfungsplan Technische Informatik aufgenommen werden.

Das Praktikum kann ebenfalls im Rahmen der Fachprüfung Technische Informatik in den Prüfungsplan mit 4 SWS aufgenommen werden.

Schriftliche Voranmeldung bei Frau Reimold, Lehrstuhl Prof. Rosenstiel

cs 1602 ww6

wgs 06-02

Client/Server Systeme

Beispiele für Prüfungsfragen

Unterschied zwischen Sockets und Ports.

Überblick über die Struktur eines einfachen Socket Programms.

Wie findet ein Socket Programm seinen Server?

Was ist ein RPC. Ist er (normalerweise) asynchron oder synchron?

Funktion der Stubs und Skeletons.

Problem des Copy by Restore.

Was macht man, wenn der Klient eine Little Endian und der Server eine Big Endian Datendarstellung verwendet. Bei Sockets? Beim RPC? Was sind XDR und ASN.1? Treten die hier angesprochenen Probleme auch bei CORBA auf? Bei Java RMI? Wodurch unterscheidet sich ASCII von EBCDIC?

Unterschied zwischen symmetrischen (z.B. DES) und asymmetrischen (z.B. RSA) Chiffres.

Aufgabe eines Key Distribution Servers. Konzept des Kerberos Authentifizierung Prozesses.

Was sind Ipsec, Secure Socket Layer und Pretty Good Privacy?

Wie funktioniert eine digitale Unterschrift?

Was ist ein Message Digest? Wozu wird es benötigt?

Was sind Access Control Listen und Capabilities?

Was ist ein Firewall und eine Demilitarized Zone? Unterschied zwischen einem Schicht 3 und einem Schicht 7 Firewall.

Unterschied zwischen einem Namensdienst (z.B. DNS) und einem Verzeichnisdienst (z.B. X.500, LDAP). Gibt es soetwas auch unter CORBA? Unter Java RMI ?

Was sind die ACID Eigenschaften bei der Transaktionsverarbeitung? Können Transaktionen parallel (multithreaded) verarbeitet werden? Probleme?

Was sind Stored Procedures, im Gegensatz zu einem Transaktionsmonitor?

Warum brauchen wir Locks (Sperrern) in der Transaktionsverarbeitung?

Was sind „Logical Units of Work“ (LUW) und Sperrpunkte (SyncPoints)?

2-Phase Commit Protokoll. Was ist es? Warum braucht man es?

Können Sie mit den Begriffen CICS (Customer Information Control System) und 3270 Protokoll etwas anfangen?

Was ist ABAP/4? Unterschied in der Vorgehensweise bei der Speicherung der Daten zwischen dem CICS und dem SAP/R3 Transaktionsmonitor.

Um einen komplexen Bildschirminhalt unter SAP R/3 aufzubauen, braucht man nur wenige Kbyte an Daten zwischen Server und Klienten zu übertragen. Funktion des SAP GUI Prozesses?

Was sind Conversational und Pseudo-conversational Transactions? Was ist eine Session? Wie wird der State einer pseudoconversationalen Transaktion gehalten?

Der RPC arbeitet (normalerweise) synchron. Sind auch asynchrone Client/Server Systeme denkbar? (Stichwort:Message oriented Middleware). Unterschied zu e-Mail?

Wie spielen Servlets und Java Server Pages zusammen um HTML Bildschirminhalte zu erzeugen. Ist ein Servlet eine Java Klasse? Eine JSP? Kann es mit einer URL aufgerufen werden?

Was ist ein Web Application Server? Was sind Enterprise Java Beans? Unterschied zwischen Entity und Session Beans? Wie speichern letztere ihre Daten?

Wie verteilt ein Web Application Server seine Last auf mehrere Prozessoren?

Welche Probleme will man mit CORBA lösen? Was ist die Aufgabe eines ORB? Aufgabe des CORBA IDL Compilers. Warum braucht man ihn? Was ist überhaupt eine „Interface (Schnittstelle)“? Aufgabe des Schnittstellen (Interface) Repositories und des Implementation Repositories.

Alternativen zu CORBA. Haben CORBA, RMI und DCOM ähnliche Funktionen? Welches Problem versucht man mit RMI over IIOP zu lösen?

Gibt es außer TCP/IP noch andere Transport Protokolle? Was ist SNA?

Client/Server Systeme unter OS/390. Können Java Programme auf CICS zugreifen? CORBA Programme? Der CICS Basic Mapping Support hat eine begrenzte Funktionalität. Was sind die Alternativen (COMMAREA, CICS Transaction Gateway?)

Was sind Connectoren, CommonConnector Framework und Java Connection Architecture?

Studienarbeiten Diplomarbeiten

siehe auch http://www-ti.informatik.uni-tuebingen.de/~spruth/index_de.html

- **Entwicklung neuer Aufgaben für das Client/Server Praktikum. Schwerpunkt Web Transaktionen**
- **Erstellen eines WebSphere Web Application Server Performance Modells für unterschiedliche Systemkonfigurationen (Win 2000, Linux, OS/390).**
- **Portierung existierender Linux Anwendungen auf OS/390 Unix System Services**
- **Weitere Themen im Umfeld OS/390-Internet Integration, Enterprise Java Beans**

Arbeiten können am Lehrstuhl Prof. Rosenstiel oder extern bei Partnern in der Wirtschaft angefertigt werden (z.B. IBM Laboratorien Böblingen).

Kontaktaufnahme:

**spruth@informatik.uni-tuebingen.de
Tel.: 0172-8051-485**

oder über den Lehrstuhl Prof. Rosenstiel / Frau Reimold