Internetanwendungstechnik (Übung)

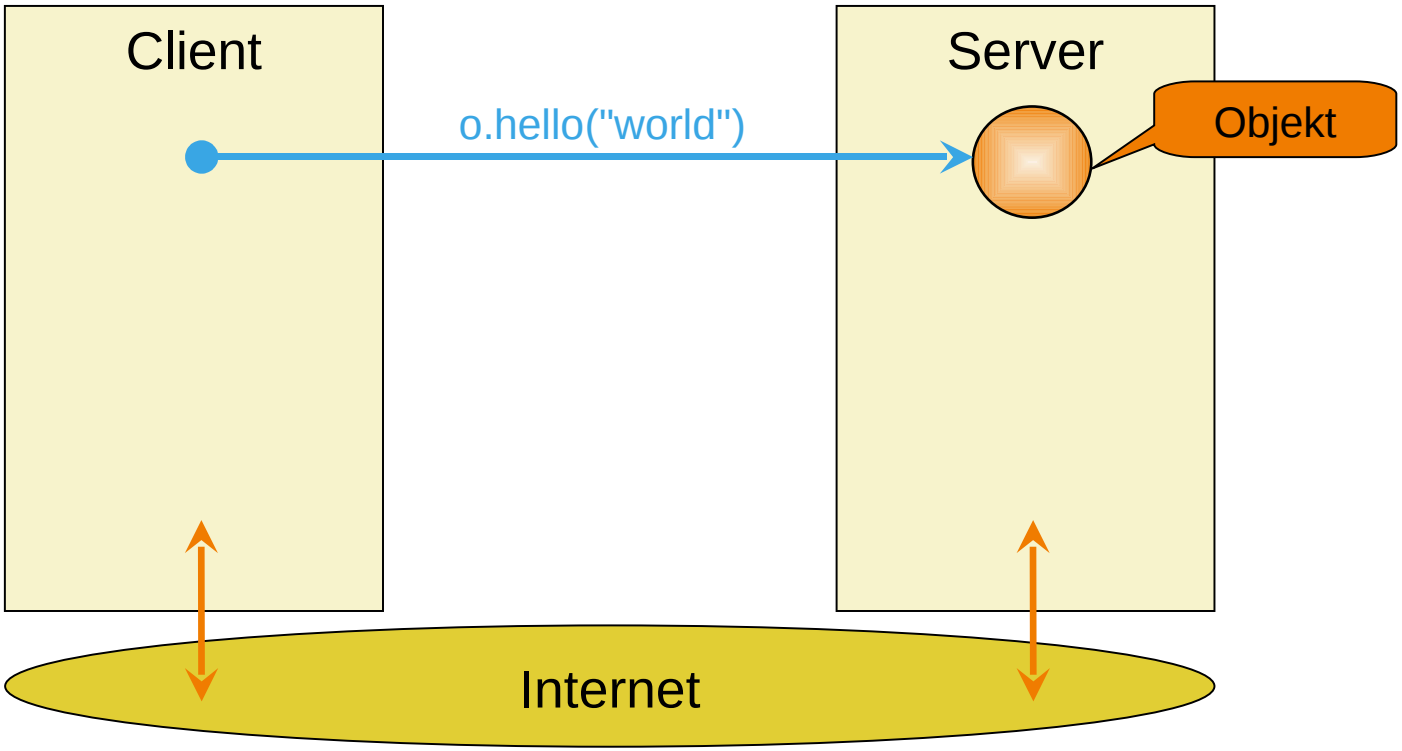
JacORB

S. Bissell, G. Mühl

Technische Universität Berlin
Fakultät IV – Elektrotechnik und Informatik
Kommunikations- und Betriebssysteme (KBS)
Einsteinufer 17, Sekr. EN6, 10587 Berlin

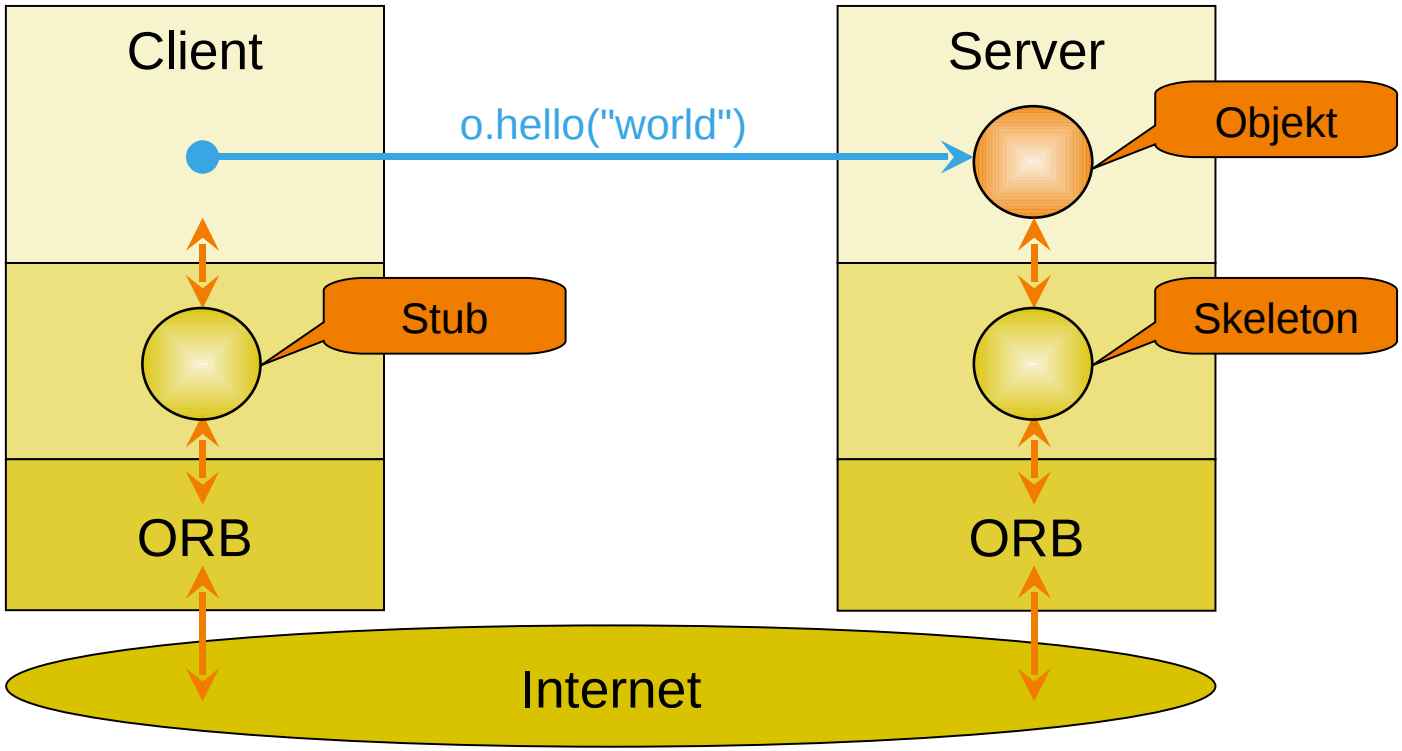
Motivation

- Entwicklung verteilter objektorientierte Anwendungen in heterogenen Umgebungen
- Methoden-/Objektaufrufe als Kommunikationsprimitive



CORBA

- Common Object Request Broker Architecture (CORBA)
- Verteilungsplattform für Objekte



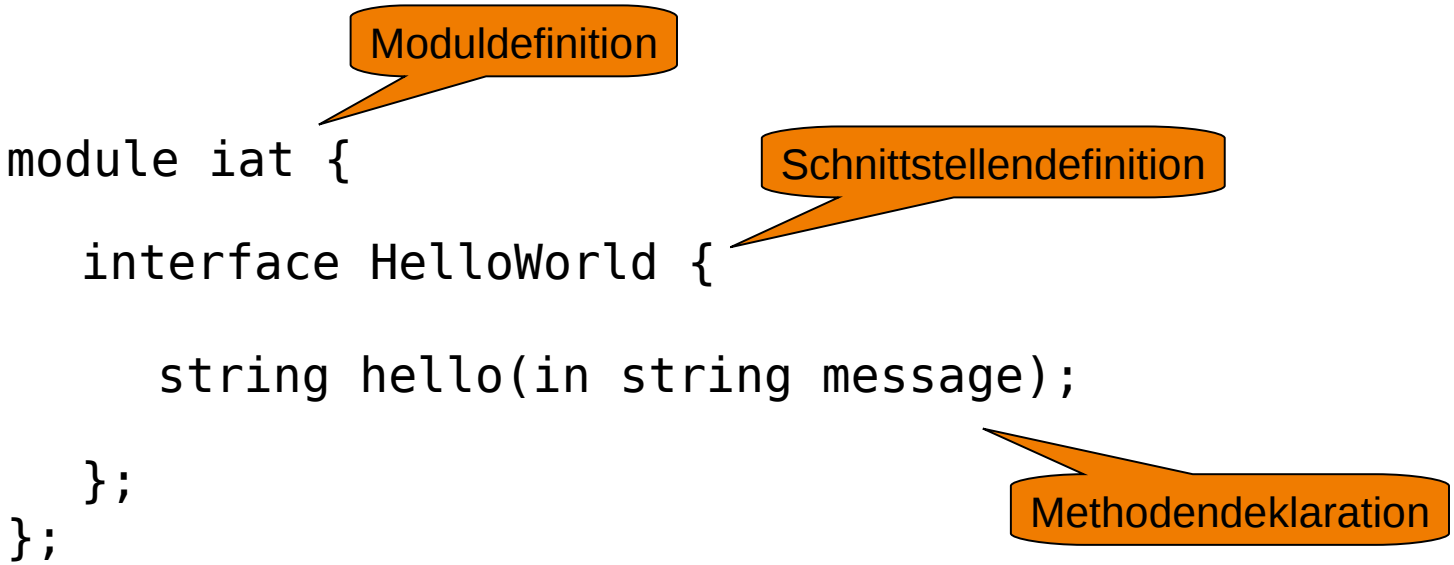
Entwicklungsschritte

2. Schnittstellen in CORBA IDL definieren
3. Mittels IDL-Compiler Stubs und Skeletons generieren
4. Implementieren des Servants (d.h. der Schnittstelle)
5. Implementieren des Servers
6. Implementieren des Clients
7. Kompilieren
8. Testen

Definition der Schnittstelle

- Interface Definition Language (IDL)
  - Sprachunabhängig
  - Plattformunabhängig
  - Rein deklarativ
  - Bietet keine Implementierung

Beispiel HelloWorld.idl




```
module iat {

   interface HelloWorld {

      string hello(in string message);

   };
};
```

Stub und Skeleton

- Erzeugung mit Hilfe des IDL Compilers
  - `$ idl HelloWorld.idl`

- Erzeugt folgende Java-Dateien im Verzeichnis iat

HelloWorld.java	Umsetzung der IDL-Definition
HelloWorldOperations.java	in Java Interface-Definitionen
HelloWorldHelper.java	Hilfsklassen (Narrowing, etc)
HelloWorldHolder.java	
HelloWorldPOA.java	Skeleton
HelloWorldPOATie.java	
_HelloWorldStub.java	Stub

Implementierung des Servants

- Servant = Implementierung des Schnittstelle
  = Objektimplementierung
- Bindung des Servants an Skeleton
  - „POA“-Ansatz
    - Bindung durch Vererbung
    - Direktes Ableiten von zugehöriger Skeleton-Klasse (POA-Klasse)
  - „POATie“-Ansatz
    - Bindung durch Delegation
    - Implementieren der zugehörigen Operations-Schnittstelle
    - Wrappen der Implementierung durch zugehörige POATie-Klasse
    - Notwendig in Sprachen mit Einfachvererbung

Beispiel HelloWorldImpl.java

Ableiten vom Skeleton

```
import iat.HelloWorldPOA;

public class HelloWorldImpl extends HelloWorldPOA {

  public String hello(String message){
    System.out.println(
      "Hello mit " + message + " aufgerufen“);
    return "Hello " + message + "!";
  }

}
```

Beispiel HelloWorldPOATieImpl.java

Implementieren der Operations-Schnittstelle

```
import iat.HelloWorldOperations;

public class HelloWorldPOATieImpl extends FooBar
                              implements HelloWorldOperations {

  public String hello(String message) {
    System.out.println(
      "Hello mit " + message + " aufgerufen“);
    return "Hello " + message + "!";
  }

}
```

Implementierung des Servers

- Aufgaben des Servers
  - Objekt HelloWorld erzeugen
  - Objekt HelloWorld aktivieren
  - Referenz für Client zur Verfügung stellen
    - IOR in Datei schreiben oder
    - Objekt beim Naming Service registrieren

Beispiel Server.java

```
import iat.HelloWorldPOATie;
public class Server {
  public static void main(String args[]){
    org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);
    org.omg.PortableServer.POA poa =
      org.omg.PortableServer.POAHelper.narrow(
        orb.resolve_initial_references("RootPOA"));
    poa.the_POAManager().activate();
    HelloWorldImpl impl = new HelloWorldImpl();
    HelloWorldPOATie impl = new HelloWorldPOATie(
                                  new HelloWorldOperationsImpl());
    poa.activate_object(impl);
    org.omg.CORBA.Object obj = poa.servant_to_reference(impl);
    // obj beim Naming Service registrieren
    String ior = orb.object_to_string(obj);
    // alternativ ior in Datei schreiben
    orb.run();
} }
```

ORB initialisieren

POA beschaffen und aktivieren

POA-Ansatz

POATie-Ansatz

Objektreferenz verfügbar machen

Implementierung des Clients

- Aufgaben des Clients
  - Objektreferenz beschaffen
    - IOR aus Datei lesen
    - Naming Service nach Referenz fragen
  - Referenz auflösen
  - Methode aufrufen

Beispiel Client.java

```
import iat.HelloWorld;
import iat.HelloWorldHelper;

public class Client {
  public static void main(String[] args) {

    org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);

    // ior aus Datei laden

    org.omg.CORBA.Object obj = orb.string_to_object(ior);

    // alternativ obj von Name Service beschaffen

    HelloWorld hw = HelloWorldHelper.narrow(obj);

    System.out.println(hw.hello("world"));

  }
}
```

IOR in Objektreferenz wandeln

Typ-Cast

Methodenaufruf

Kompilieren und Testen

- Kompilieren

```
$ mkdir classes
$ javac –d classes iat/*.java *.java
```

- Testen
  - Name Service starten

    ```
    $ ns
    ```

  - Server starten

    ```
    $ jaco -cp classes Server [server.ior]
    ```

  - Client starten

    ```
    $ jaco -cp classes Client [server.ior]
    ```

Alternativ Datei zum Speichern der IOR angeben

Name Service

- Referenz des Name Services anfordern

```
org.omg.CosNaming.NamingContextExt nc =
  org.omg.CosNaming.NamingContextExtHelper.narrow(
    orb.resolve_initial_references("NameService"));
```

- Servant registrieren

```
nc.bind( nc.to_name("HelloWorld"), obj );
```

- Namen auflösen

```
HelloWorld hw = HelloWorldHelper.narrow(
  nc.resolve( nc.to_name("HelloWorld")));
```

Voraussetzungen

- Java Software Development Kit (SDK)
  - Version 1.3 oder höher
  - http://java.sun.com/javase/
- Apache Ant
  - Nur zum (selbst) Kompilieren benötigt
  - Version 1.5 oder höher
  - http://ant.apache.org/
- JacORB
  - Frei CORBA-Implementierung für Java
  - Entstanden an der FU Berlin
  - Aktuelle Version 2.3
  - http://www.jacorb.org/

Umgebungsvariablen

- `JAVA_HOME`, `ANT_HOME` und `JACORB_HOME` verweisen auf entsprechende Installationsverzeichnisse
- Suchpfad enthält entsprechende bin-Verzeichnisse → `{$JAVA_HOME,$ANT_HOME,$JACORB_HOME}/bin`
- JAVA Classpath enthält `jacorb.jar`
- Setzen unter Unix (z.B. in der .bashrc)

```
$ export JACORB_HOME=/opt/JacORB
$ export PATH=$PATH:${JACORB_HOME}/bin
$ export CLASSPATH=".;${JACORB_HOME}/lib/jacorb.jar"
```

- Setzen unter Windows XP

  Systemsteuerung | Leistung und Wartung | System | Erweitert | Umgebungsvariablen | Neu bzw. Bearbeiten

Konfiguration

- Konfiguration erfolgt durch Datei jacorb.properties

```
$ cp ${JACORB_HOME}/etc/jacorb_properties.template
     ~/jacorb.properties
```

- Konfiguration des Name Service
  - Abschnitt „Initial Reference Configuration“ in jacorb.properties

    `ORBInitRef.NameService=file:/somewhere/NS_Ref`
  - Abschnitt „Name Service Configuration“ in jacorb.properties

    `jacorb.naming.ior_filename=/somewhere/NS_Ref`

Fragen?