Freie Universität Berlin
Klaus-Peter Löhr

9 CORBA

9.1 Konzepte

Plattformen/Middleware für objektorientierte verteilte Anwendungen

- **Java RMI**:
  Fernaufrufe nur zwischen JVM-Komponenten

- **Microsoft .NET**:
  Fernaufrufe zwischen CLR-Komponenten

- **CORBA**:
  Fernaufrufe unabhängig von Sprachen, Systemen, Herstellern

CORBA

- *"Common Object Request Broker Architecture"*
- spezifiziert von der *Object Management Group (OMG,1991)*
  (Konsortium „aller bekannten" IT-Firmen)
- als „**ORB**" *(Object Request Broker)* vielfach implementiert
  - kommerziell und nichtkommerziell
- verbreitet für die Anbindung von Altsoftware *(legacy software)*
- für Neuprojekte verdrängt durch Web-basierte Techniken

Elemente von CORBA

- **IDL** *(Interface Definition Language)* für sprachunabhängiges Objektmodell und Beschreibung von Schnittstellen
- **Sprachanbindung** *(IDL-language mapping):*
  Zuordnung zwischen IDL und den Sprachelementen der jeweiligen Programmiersprache (hauptsächlich Typsystem)
- **Namensdienst** *(Name Service):*
  ähnlich RMI-Register, aber mit hierarchischem Namensraum
- **IIOP** *(Internet Inter-ORB Protocol):*
  analog zu Javas JRMP, aber sprachunabhängig, IDL-bezogen
- ... weitere *„Services", „Facilities", ...*

9.2. IDL

... hat oberflächliche Ähnlichkeit mit „gängigen" Programmiersprachen

- **`module`** umfasst die Beschreibung von (eventuell mehreren) Schnittstellen, Datentypen, Ausnahmen, ...
- **`interface`** beschreibt eine Objekt-Schnittstelle, bestehend aus Objektattributen, Signaturen von **Operationen**, ...
- **`attribute`** beschreibt ein Objektattribut, auf das mit *getter/setter*-Methoden zugegriffen werden kann (!)
- **`exception`** wird wie Verbundtyp vereinbart; in Signatur: **`raises`**

```
module counter {  // simple example

  struct Info {
    long value;
    long invoc;   // number of inc invocations
  };

  interface Counter {
    readonly attribute long value;
    void inc(in long value);
    void addTo(inout long clientValue);
    Info getInfo();
  };

};
```

Einige Besonderheiten

- Objektorientierung? Nur Schnittstellen-Vererbung.
- Schnittstellentypen sind *Verweistypen*
  - alle anderen Typen *nicht.*
- *Parametermechanismen:* Wert/Ergebnisübergabe
  - bei Schnittstellentyp Übergabe eines Fernverweises.
- **`valuetype`** ähnlich wie **`interface`**
  - aber *kein* Verweistyp, also Übergabe einer Kopie
  (aber keine Unterstützung von Code-Mobilität).
- Signatur mit Zusatz **`oneway`** (falls **`void`** und kein **`out`**):
  asynchrone Ausführung mit höchstens-einmal-Semantik.

Sprachanbindung am Beispiel Java:

IDL	Java
module	**package**
interface	**interface**
(nichts)	**class**
Operation	Methode
raises	**throws**
in-Parameter	Argument
out, **inout**	(nichts!)
Ergebnis	Ergebnis

IDL	Java
`octet`	`byte`
`char`	`char`
`long`	`int`
`long long`	`long`
`float`	`float`
`double`	`double`
`...`	`...`
struct	**final class**
sequence	`[]` (Feld)

! Das alles geht nicht ohne gewisse semantische Brüche !

9.3 Praxis

Für die Spezifikation `counter` aus 9.2 Implementierung fernaufrufbarer `Counter`-Objekte am Beispiel Java:

```
package counter;
import  org.omg.CORBA.*;
import  org.omg.CosNaming.*;
import  org.omg.PortableServer.*;

public class CounterImpl extends CounterPOA {
        int value = 0;
        int addCount = 0;
public int value() { return value; }
public void inc(int i) {
        value += i;
        addCount++;
        }
..........
```

Freie Universität Berlin

Klienten-Code für einen Test
- der Einfachheit halber im gleichen Paket:

```
package counter;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;

public class CounterClient {
public static void main(String[] arg) throws Exception {
       ORB orb = ORB.init(arg, null); // ORB initialization
.....
```

Sprachspezifischer **IDL-Übersetzer** *(IDL compiler)* erzeugt aus gegebenem IDL-Text Code in verschiedenen Dateien:

```
$  idlj –fall counter.idl
```

Der IDL/Java-Übersetzer erzeugt aus `counter.idl` diverse Klassen/Schnittstellen im Paket `counter`, u.a. Klassen für *Stub (client stub)* und *Skeleton (server stub)*, gespeichert in Dateien `counter/*.java` .

```
$  javac counter/*.java
```

alles übersetzen für einfachen lokalen Test

Lokaler Test:

(Der Namensdienst wird durch den *ORB Daemon* `orbd` - mit Standard-Port 900 - realisiert.)

```
$ orbd -ORBInitialPort 1050 &
[1] 9565
$ java counter.CounterImpl -ORBInitialPort 1050 &
[2] 9569
$ java counter.CounterClient -ORBInitialPort 1050
.....
```

Interoperabilität

- Java RMI unterstützt Interoperabilität mit CORBA:
  - „*RMI over IIOP*" (statt über JRMP)
  - Pakete `javax.rmi`, `javax.rmi.CORBA`
  - http://docs.oracle.com/javase/1.4.2/docs/guide/corba/

- CORBA + .NET + RMI: **IIOP.NET** (ETH Zürich)
  - gemeinsame Basis ist .NET Remoting
  - http://iiop-net.sourceforge.net/

Beurteilung

- CORBA ist mühsamer zu benutzen als Java RMI oder .NET Remoting. CORBA ist *komplizierter* als es zur Lösung des Heterogenitätsproblem tatsächlich nötig wäre.

- Die *Sprachanbindungen* haben ihre Tücken und sind in der praktischen Anwendung fehlerträchtig.

- Weitere *Schwächen* (Versionierung, Fernaufrufprotokoll, Verteilungsabstraktion, Sicherheitsprobleme, ...)

- *„Design by committee"* ist die Ursache vieler dieser Schwächen.

Zusammenfassung

- CORBA erlaubt Fernaufrufe zwischen verschiedenen Sprachen: bei der Entwicklung eines Klienten muss keinerlei Rücksicht darauf genommen werden, in welcher Sprache der Server implementiert ist.
- „Objektorientierte" Schnittstellenbeschreibungssprache IDL ist unabhängig von den verwendeten Programmiersprachen. Sprachanbindung hat aber notgedrungen Schwächen.
- Schwerfälliger in der Benutzung als RMI oder .NET Remoting.
- Für heterogene verteilte Anwendungen wird heute eher Web-basierte Middleware eingesetzt.

Quellen

OMG: *CORBA 3.0 Specification. IDL Language Mappings.*
www.omg.org/technology/documents/corba_spec_catalog.htm
www.omg.org/technology/documents/spec_catalog
www.omg.org/technology/documents/idl2x_spec_catalog.htm

G. Brose et al.: *JacORB.* www.jacorb.org
(Freier ORB in Java - hier am Institut entwickelt!)

G. Brose, A. Vogel, K. Duddy: *Java Programming with CORBA.* Wiley 2001

J. Farley, W. Crawford: *Java Enterprise in a Nutshell* (3. ed.). O'Reilly 2005

M. Henning: *The Rise and Fall of CORBA.* Comm. of the ACM 51.8,
August 2008, pp. 53-57