Team: TEAM 01, Falco Winkler (FW), Daniel Schruhl (DS)

Aufgabenteilung:

- IDL Compiler
- Namensdienst
- mware_lib Library

Quellenangaben:

- Aufgabe 4, 11.06.2017, C. Klauck & H. Schulz:
  http://users.informatik.haw-hamburg.de/ schulz/pub/Verteilte-Systeme/AI5-VSP/Aufgabe4/

Bearbeitungszeitraum:

- 11.06.2017 4 Stunden (DS)

Aktueller Stand:

- IDL Compiler begonnen
- Namensdienst
- mware_lib Library

Änderung des Entwurfs:

- Keine Änderungen

1 Einführung und Ziele

Es soll eine einfache objektorientierte Middleware entworfen werden, die Methodenaufrufe eines entfernten Objektes ermöglicht.
Zur Orientierung gilt hierbei die CORBA Architektur. Genauer soll hier ein ORB zur Verfügung gestellt werden, der es ermöglicht Methoden von entfernten Objekten aufzurufen.
Zur Abstraktion und Beschreibung der Schnittstellen der Objekte soll eine IDL verwendet werden. Diese IDL wird dann zur Erzeugung von Klassen- und Methodenrümpfen verwendet.
Außerdem beinhaltet der ORB einen Namensdienst, der Objektreferenzen in einem Netz mit Namen finden kann.
Die Middleware an sich soll durch eine Library abstrahiert und verwendbar sein.

1.1 Randbedingungen

Der Namensdienst soll auf einem entfernten Rechner unabhängig von der Middleware Library lauffähig sein. Der Port muss zur Laufzeit einstellbar sein.
Der IDL-Compiler soll in einem Package oder einer `.jar` Datei zur Verfügung gestellt werden. Der Compiler soll folgende IDL Typen unterstützen:

- module (keine Schachtelung, 1 Modul pro Datei)
- class (nicht als Parameter oder Returnwert, keine Schachtelung)
- int
- double
- string

Ein Beispiel:

```
module math_ops {
  class Calculator {
   double add(double a, double b);
   string getStr(double a);
 };
};
```

Die Middleware Library soll in einem Package `mware_lib` zusammengefasst werden.
Wenn eine Serverapplikation während eines entfernten Methodenaufrufes eine RuntimeException wirft, soll diese an den Aufrufer weitergeleitet werden.
Es soll möglich sein, dass zwei oder mehrere Klienten die selbe Objektreferenz zeitgleich nutzen wollen. Das soll innerhalb der Middleware nicht zu Deadlocks führen.

1.2 Kontextbegrenzung

Die Implementierung soll in Java vorliegen.
Die Behebung von Deadlocks in den Anwendungen ist nicht Aufgabe der Middleware.

2 Gesamtsystem

Das Gesamtsystem besteht aus drei Subsystemen, die unabhängig voneinander lauffähig sind. Dabei sind das NameService System und das IDL-Compiler System Applikationen, die über die Kommandozeile gestartet werden können.
Die Middleware Library (mware_lib) ist eine Bibliothek, die in der jeweiligen Applikation, die die Middleware verwenden soll eingebunden werden muss. Die Middleware Library ist also nicht direkt über die Kommandozeile startbar.

2.1 Bausteinsicht

2.2 Laufzeitsicht

3 Subsysteme und Komponenten

3.1 NameService

3.1.1 Aufgabe und Verantwortung

Der NameService bildet Namen auf Objektreferenzen ab. Er wird verwendet, um Objekte anhand ihres Namens zu finden und anzusprechen und um Objekte anzumelden.

3.1.2 Schnittstelle

3.1.3 Entwurfsentscheidungen

Der Port, an dem der NameService läuft ist zur Laufzeit einstellbar. Das geschieht über den Startparameter.

3.1.4 Konfigurationsparameter

- Port des NameServices

3.2 IDL Compiler

3.2.1 Aufgabe und Verantwortung

Der IDL Compiler hat die Aufgabe, eine gegebene Modulbeschreibung aus einer Datei im .idl - Format einzulesen und die der Beschreibung entsprechende abstrakte Java-Klasse als Java-Code zu generieren. Die generierte Klasse wird auf Client und Serverseite eingebunden und zu einer konkreten Klasse abgeleitet. Instanzen davon werden als Proxy - Objekte für CORBA - Aufrufe verwendet.

3.2.2 Schnittstelle

`Compiler.main(String[] args)`
Ueber die main – Methode der Klasse Compiler kann der Compiler ausgefuehrt werden. Als erstes Argument muss der Pfad der .idl–Datei, als zweites Argument der Pfad zur Ausgabedatei angegeben werden.

3.2.3 Entwurfsentscheidungen

Die Methoden zur Übersetzung der IDL - Beschreibung werden abstrakt in einem Interface definiert. Die Eigentliche Übersetzung nimmt IDLToJavaTranslator (die Implementierung des Interfaces) vor. So ist der Übersetzungsprozess logisch von der Syntax der zu übersetzenden Sprache getrennt, und die Testbarkeit der einzelnen Vorgänge gegeben.

3.2.4 Konfigurationsparameter

- Pfad zur Eingabedatei
- Pfad zur Ausgabedatei (existierende Dateien werden überschrieben)