JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Laborprotokoll Dezentrale Systeme

CORBA Java / C++

Systemtechnik Labor
4CHITT 2015/16, GruppeB

Philipp Kogler

Version 0.2
Note:
Begonnen am 31. März 2016
Betreuer: BORKO Michael
Beendet am 31. März 2016

Inhaltsverzeichnis

1 Einführung

Verteilte Objekte haben bestimmte Grunderfordernisse, die mittels implementierten Middlewares leicht verwendet werden können. Das Verständnis hinter diesen Mechanismen ist aber notwendig, um funktionale Anforderungen entsprechend sicher und stabil implementieren zu können.

1.1 Ziele

Diese Übung gibt eine einfache Einführung in die Verwendung von verteilten Objekten mittels CORBA. Es wird speziell Augenmerk auf die Referenzverwaltung sowie Serialisierung von Objekten gelegt. Es soll dabei eine einfache verteilte Applikation in zwei unterschiedlichen Programmiersprachen implementiert werden.

1.2 Voraussetzungen

- Grundlagen Java, C++ oder anderen objektorientierten Programmiersprachen
- Grundlagen zu verteilten Systemen und Netzwerkverbindungen
- Grundlegendes Verständnis von nebenläufigen Prozessen

1.3 Aufgabenstellung

Verwenden Sie das Paket ORBacus oder omniORB bzw. JacORB um Java und C++ ORB-Implementationen zum Laufen zu bringen.

Passen Sie eines der Demoprogramme (nicht Echo/HalloWelt) so an, dass Sie einen Namingservice verwenden, welches ein Objekt anbietet, das von jeweils einer anderen Sprache (Java/C++) verteilt angesprochen wird. Beachten Sie dabei, dass eine IDL-Implementierung vorhanden ist um die unterschiedlichen Sprachen abgleichen zu können.

Vorschlag: Verwenden Sie für die Implementierungsumgebung eine Linux-Distribution, da eine optionale Kompilierung einfacher zu konfigurieren ist.

2 Corba Allgemein (Overview)

2.1 Was ist CORBA [1] [2]

CORBA (**C**ommon **O**bject **R**equest **B**roker **A**rchitecture) vereinfacht das Erstellen verteilter Anwendungen und soll das Aufrufen externer Methoden ermöglichen bzw. vereinfachen.

Der große Vorteil von CORBA ist die Plattformunabhängigkeit und ist somit, im Gegensatz zu anderen Umsetzungen, nicht an eine Spezielle Umgebung gebunden. CORBA setzt darauf, dass die Hersteller bzw. Communities, auf der Grundlage der Spezifikation eigene Object-Request-Broker Implementierungen erstellen und weiter verbessern. Deshalb können Hersteller Ihre Implementierungen für mehrere Programmiersprachen und auch unterschiedliche Betriebssysteme anbieten.

Die gemeinsame Spezifikation ermöglicht dann die Kommunikation von Anwendungen untereinander, die mit unterschiedlichen Programmiersprachen erstellt worden sind, verschiedene ORBs nutzen und auf verschiedenen Betriebssystemen und Hardwareumgebungen laufen können.

2.2 Wie funktioniert CORBA [1] [2]

Mithilfe von der *Interface Definition Language (IDL)* können formale Spezifikationen der Schnittstellen (Datentypen und Methodensignaturen), die ein Objekt für remote oder lokale Zugriffe zur Verfügung stellen, umgesetzt werden.

Damit das ganze Prinzip funktioniert müssen diese definierten Schnittstellen (in Java Interfaces) für alle anderen Programmiersprachen umgesetzt werden. Dafür müssen diese nun von dem entsprechenden IDL-Compiler in äquivalente Beschreibungen der Schnittstellen kompiliert werden.

Ebenfalls wird Quellcode, welcher zu der benutzten ORB-Implementierung passt, erstellt. Deiser Quelltext enthält, wie wir bereits von *Remote Method Invocation* kennen, die Implementierung des Skelletons bzw. Stubs für Callback am Client usw. Durch dieses *Vermittler-Pattern* erscheinen remote Aufrufe fast so einfach wie lokale Aufrufe und verbirgt somit die Komplexität der damit verbundenen Netzwerkkommunikation.

Bei unserem Beispiel verwenden wie die Java Implementierung **`jackorb [4]`** und die C++/Python Implementierung **`omniOrb [1]`**

3 Vorbereitung [3]

Damit wir die **`omniOrb [1]`** bzw **`jackorb [4]`** Implementationen verwenden können müssen diese zuerst gebuildet bzw. kompiliert werden.

Der große Vorteil ist, dass das Builden / Kompilieren für die entsprechende Plattform durchgeführt wird und somit auf jeden Fall die optimale Leistung bzw. Effizienz herausholen kann.

Bei der Kompilierung ist darauf zu achten in welcher Programmiersprache der Programmcode geschrieben wurde und welcher Compiler von dem Programmierer bevorzugt wird.
[!TIPP IMMER ALLE README LESEN!]

3.1 Kompilieren von `omniOrb [1]`

```
1 apt-get install
```

Listing 1: Implizite Transaktion [**?**]

4 Ergebnisse

Literatur

[1] http://omniorb.sourceforge.net/.

[2] https://de.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture.

[3] https://wiki.ubuntuusers.de/Programme_kompilieren/.

[4] http://www.jacorb.org/.

Tabellenverzeichnis

Listings

Abbildungsverzeichnis