CORBA

Systemtechnik Labor
4CHIT 2015/16

Boban Jevtic

Version 0.1

Note:

Betreuer: Michael Borko

Begonnen am 18. März 2016

Beendet am 1. April 2016

Inhalt

Einführung:

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.

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.

Voraussetzungen:

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

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.

Ergebnisse:

Erklärung:

Allgemein:

CORBA steht für „Common Object Request Broker Architecture“. Die dahinter ist, dass man innerhalb der Sprache andere Objekte benutzen kann. CORBA funktioniert auf unterschiedlichen Systemen. Um dieses Tool zum Laufen zu bringen, wäre eine Möglichkeit, dass man verschiedene Systeme miteinander verbindet und beispielsweise mit unterschiedlichen Programmiersprachen arbeitet.

Es gibt folgende Arten von Services:

- NamingService
- SecurityService
- NotificationService
- PersistentStateService

Zuerst muss das Objekt mithilfe von NamingService. identifiziert werden. Die Idee dahinter ist, dass man jegliche Art von NamingService verwenden kann. In Naming Service wird festgestellt, welcher Port auf welchem Server läuft. In der CORBA Registry müssen die Namen im Context eindeutig sein. Dabei ist hier die Objektzugehörigkeit sehr bedeutsam.

Prinzip von CORBA:

Das Prinzip ist ähnlich, wie bei RMI. Auf der linken Seite befindet ein der Client und auf der rechten Seite befindet sich ein Server. Unten gibt es eine Verbindungschnittstelle und zwischen den Ebenen gibt es Interfaces. Es wird ein Interface benötigt, wenn ich beispielsweise als Client etwas aufrufen will, das irgendwo eine absolute Objektreferenz hat. ORB kümmert sich um die Netzwerkschnittstelle.

Request Broker basiert auf TCP/IP und ist nichts anderes als eine implementierte Schnittstelle. ORB weist, dass es eine Referenz gibt und ORB schaut sich diese Referenz genau an. ORB weist, dass es einen Port und eine IP-Adresse gibt.

Nun wird die Anfrage von ORB an den Object Adapter übergeben. Der Object Adapter kennt nun die Anfrage von ORB und er kümmert sich um den Aufruf. Der Object Adapter kann auch die Perstierung durchführen. Man braucht aber auch eine Referenz von ORB (Object Request Broker).

Der Portable Object Adapter (POA) kümmert sich darum, dass eine Instanz zur Verfügung steht. POA kümmert sich auch um die Interaktion zwischen ORB und Distributed Objects.

IDL:
Mithilfe der IDL kann ich egal in welcher CORBA Implementierung herangehen und den Compiler nehmen, der den Stub Code und Skeleton Code erzeugt.

IDL (Interface Definition Language) ist eine deskriptive Sprache, die man bei der CORBA Übung verwenden muss. IDL Stub und IDL Skeleton wird von IDL-Compiler zur Verfügung gestellt.

Man braucht bestimmte Elemente (Module im IDL) wie Namespace. Module in Java können Packages sein. Module definieren Namensräume:

- Konstanten
- Datentypen
- Exceptions
- CORBA Objekte

Dieses IDL hilft uns unabhängig von den Programmiersprachen zu arbeiten. Die Idee dahinter ist, dass man Objekte einmal schreiben muss und diese für die entsprechende Sprache übersetzen lassen kann.

Folgende Voraussetzungen müssen erfüllt werden, um Java und C++ ORB-Implementation zum Laufen zu bringen

Zuerst müssen folgende Pakete runtergeladen werden:

omniORB:

Das Paket omniORB kann mit folgendem Befehl heruntergeladen werden. Dieses Packet muss man zunächst entpacken.

```
root@debian:~# sudo wget https://sourceforge.net/projects/omniorb/files/omniORB/omniORB-4.2.1/
--2016-03-29 18:22:06--  https://sourceforge.net/projects/omniorb/files/omniORB/omniORB-4.2.1/
Resolving sourceforge.net (sourceforge.net)... 216.34.181.60
Connecting to sourceforge.net (sourceforge.net)|216.34.181.60|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 40741 (40K) [text/html]
Saving to: 'index.html'
```

Bevor die Kompilierung konfiguriert wird, sind folgende Programme zu installieren:

Python, gcc, openjdk7, make

Viele Programme wurden bereits schon installiert. Python wurde schon installiert.

```
root@debian:~/omniORB-4.2.1/build# sudo apt-get install python
Reading package lists... Done
Building dependency tree
Reading state information... Done
python is already the newest version.
The following packages were automatically installed and are no longer required:
  python-defusedxml python-soappy python-wstools
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 233 not upgraded.
```

Gcc wurde schon installiert.

```
root@debian:~/omniORB-4.2.1/build# sudo apt-get install gcc
Reading package lists... Done
Building dependency tree
Reading state information... Done
gcc is already the newest version.
The following packages were automatically installed and are no longer required:
  python-defusedxml python-soappy python-wstools
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 233 not upgraded.
```

Java SDK 7 wurde schon installiert.

```
root@debian:~/omniORB-4.2.1/build# sudo apt-get install openjdk-7-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
openjdk-7-jdk is already the newest version.
The following packages were automatically installed and are no longer required:
  python-defusedxml python-soappy python-wstools
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 233 not upgraded.
```

Die automatische Konfiguration wird ausgeführt. Der Befehl confiugre schaut sich das System an und bildet das Makefile.

```
root@debian:~/omniORB-4.2.1/build# ../configure
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for g++... g++
checking whether we are using the GNU C++ compiler... yes
checking whether g++ accepts -g... yes
checking how to run the C preprocessor... gcc -E
checking for ranlib... ranlib
checking for a BSD-compatible install... /usr/bin/install -c
checking whether make sets $(MAKE)... yes
checking for python... /usr/bin/python
checking for python version... 2.7
checking for python platform... linux2
checking for python script directory... ${prefix}/lib/python2.7/dist-packages
checking for python extension module directory... ${exec_prefix}/lib/python2.7/dist-packages
checking for pkg-config... /usr/bin/pkg-config
checking pkg-config is at least version 0.9.0... yes
```

Nun wird zunächst die Kompilierung durchgeführt.

```
root@debian:~/omniORB-4.2.1/build# make
making export in ./src...
make[1]: Entering directory '/root/omniORB-4.2.1/build/src'
making export in src/tool...
make[2]: Entering directory '/root/omniORB-4.2.1/build/src/tool'
making export in src/tool/omkdepend...
make[3]: Entering directory '/root/omniORB-4.2.1/build/src/tool/omkdepend'
File omkdepend hasn't changed.
make[3]: Leaving directory '/root/omniORB-4.2.1/build/src/tool/omkdepend'
making export in src/tool/omniidl...
make[3]: Entering directory '/root/omniORB-4.2.1/build/src/tool/omniidl'
making export in src/tool/omniidl/cxx...
make[4]: Entering directory '/root/omniORB-4.2.1/build/src/tool/omniidl/cxx'
../../../../bin/omkdepend -D__cplusplus -D__GNUG__ -D__GNUC__ -I/usr/include/python2.7 -DPYTHON_INCLUDE=<Python.
h> -DPYTHON_THREAD_INC=<pythread.h> -DIDLMODULE_VERSION="0x2630" -fPIC -I. -I../../../../../../src/tool/omniidl/cxx
 -I../../../../include -I../../../../../include -D__OSVERSION__=2 -D__linux__ -D__x86_64__ ../../../../../../src/to
ol/omniidl/cxx/idlc.cc ../../../../../../src/tool/omniidl/cxx/idlpython.cc ../../../../../../src/tool/omniidl/cxx/idlf
ixed.cc ../../../../../../src/tool/omniidl/cxx/idlconfig.cc ../../../../../../src/tool/omniidl/cxx/idldump.cc ../../..
/../../../src/tool/omniidl/cxx/idlvalidate.cc ../../../../../../src/tool/omniidl/cxx/idlast.cc ../../../../../../src/tool
/omniidl/cxx/idlexpr.cc ../../../../../../src/tool/omniidl/cxx/idlscope.cc ../../../../../../src/tool/omniidl/cxx/idlr
epoId.cc ../../../../../../src/tool/omniidl/cxx/idltype.cc ../../../../../../src/tool/omniidl/cxx/idlutil.cc ../../../
../../../src/tool/omniidl/cxx/idlerr.cc ../../../../../../src/tool/omniidl/cxx/lex.yy.cc ../../../../../../src/tool/omnii
```

Damit die Kompilierung erfolgreich abschlossen werden kann, muss zunächst der Befehl „make install“ durchgeführt werden.

```
root@debian:~/omniORB-4.2.1/build# make install
making install in ./src...
make[1]: Entering directory '/root/omniORB-4.2.1/build/src'
making install in src/tool...
make[2]: Entering directory '/root/omniORB-4.2.1/build/src/tool'
making install in src/tool/omkdepend...
make[3]: Entering directory '/root/omniORB-4.2.1/build/src/tool/omkdepend'
File omkdepend hasn't changed.
make[3]: Leaving directory '/root/omniORB-4.2.1/build/src/tool/omkdepend'
making install in src/tool/omniidl...
make[3]: Entering directory '/root/omniORB-4.2.1/build/src/tool/omniidl'
making install in src/tool/omniidl/cxx...
make[4]: Entering directory '/root/omniORB-4.2.1/build/src/tool/omniidl/cxx'
making install in src/tool/omniidl/cxx/cccp...
make[5]: Entering directory '/root/omniORB-4.2.1/build/src/tool/omniidl/cxx/cccp
'
File omnicpp hasn't changed.
make[5]: Leaving directory '/root/omniORB-4.2.1/build/src/tool/omniidl/cxx/cccp'
+ /usr/bin/install -c -m 0644 _omniidlmodule.so.4.2 /usr/local/lib/python2.7/dis
t-packages
+ cd /usr/local/lib/python2.7/dist-packages
+ rm -f _omniidlmodule.so.4
+ ln -s _omniidlmodule.so.4.2 _omniidlmodule.so.4
+ rm -f _omniidlmodule.so
```

Bevor man die Kompilierung startet, muss man sich das IDL-File holen.

```
root@debian:~/omniORB-4.2.1/src/examples/echo# cd ../../..
root@debian:~/omniORB-4.2.1# cd bin/
root@debian:~/omniORB-4.2.1/bin# ls
i568_linux_2.0  scripts  x86_win32
root@debian:~/omniORB-4.2.1/bin# mv i568_linux_2.0 i586_linux_2.0
root@debian:~/omniORB-4.2.1/bin# ls
i586_linux_2.0  scripts  x86_win32
root@debian:~/omniORB-4.2.1/bin# cd ../src/examples/echo/
```

In diesem echo-Ordner wird der Befehl make ausgeführt.

```
root@debian:~/omniORB-4.2.1/src/examples/echo# make
dir=../../../stub;  if [ ! -d $dir ]; then (umask 002; set -x; mkdirhier $dir); fi
rm -f ../../../stub/echo.idl
../../../bin/i586_linux_2.0/omniidl -bcxx -Wba -Wbtp -C../../../stub  ../../../idl/echo
.idl
../../../bin/i586_linux_2.0/omkdepend -D__cplusplus -D__GNUG__ -D__GNUC__ -D__OMNIORB4_
_ -I../../../stub -D_REENTRANT -I. -I. -I../../../include -D__x86__ -D__linux__ -D__OSV
ERSION__=2 eg3_clt.cc eg3_impl.cc eg2_clt.cc eg2_impl.cc eg1.cc

cd ../../../stub
make[1]: Entering directory '/root/omniORB-4.2.1/stub'
../bin/i586_linux_2.0/omkdepend -D__cplusplus -D__GNUG__ -D__GNUC__ -D__OMNIORB4__ -I..
/stub -D_REENTRANT -I. -I. -I../include -D__x86__ -D__linux__ -D__OSVERSION__=2 echoDyn
SK.cc echoSK.cc
g++ -c  -fhandle-exceptions -Wall -Wno-unused -D__OMNIORB4__ -I../stub -D_REENTRANT  -I
. -I. -I../include -D__x86__ -D__linux__ -D__OSVERSION__=2 -o echoSK.o echoSK.cc
g++: warning: -fhandle-exceptions has been renamed -fexceptions (and is now on by defau
lt)
make[1]: Leaving directory '/root/omniORB-4.2.1/stub'
+ rm -f eg1
+ g++ -o eg1 -fhandle-exceptions -Wall -Wno-unused -Wl,-rpath,../../../lib/i586_linux_2
.0 -L../../../lib/i586_linux_2.0 eg1.o ../../../stub/echoSK.o -lomniORB4 -lomnithread -
lpthread
g++: warning: -fhandle-exceptions has been renamed -fexceptions (and is now on by defau
lt)
g++ -c  -fhandle-exceptions -Wall -Wno-unused -D__OMNIORB4__ -I../../../stub -D_REENTRA
NT -I. -I. -I../../../include -D__x86__ -D__linux__ -D__OSVERSION__=2 -o eg2_impl.o eg2
 impl.cc
```

In diesem Verzeichnis sind vorgefertigte Examples zu finden.

```
root@debian:~/omniORB-4.2.1/src/examples/echo# ls
dir.mak    eg1.cc      eg2_clt.d    eg2_impl.o  eg3_impl     eg3_tieimpl.cc
dir.mk     eg1.d       eg2_clt.o    eg3_clt     eg3_impl.cc  eg3_tieimpl.o
echo.hh    eg1.o       eg2_impl     eg3_clt.cc  eg3_impl.d   GNUmakefile
echoSK.cc  eg2_clt     eg2_impl.cc  eg3_clt.d   eg3_impl.o   GNUmakefile.in
eg1        eg2_clt.cc  eg2_impl.d   eg3_clt.o   eg3_tieimpl  README
```

Bei der Ausführung tritt ein Fehler auf.

```
root@debian:~/omniORB-4.2.1/src/examples/echo# ./eg1
./eg1: error while loading shared libraries: libomniORB4.so.2: cannot open shared object file:
No such file or directory
```

Nachdem die Fehlermeldung aufgetreten ist, sind diese folgende und wichtige Kommandos Schritt für Schritt auszuführen.

```
root@debian:~/omniORB-4.2.1/src/examples/echo# mkdir /root/omniORB-4.2.1/lib
root@debian:~/omniORB-4.2.1/src/examples/echo# ln -s /root/omniORB-4.2.1/build/lib /root/omniOR
B-4.2.1/lib/i586_linux_2.0
root@debian:~/omniORB-4.2.1/src/examples/echo# ln -s /root/omniORB-4.2.1/build/lib /root/omniOR
B-4.2.1/lib/i586_linux_2.0
root@debian:~/omniORB-4.2.1/src/examples/echo# LD_LIBRARY_PATH=/root/omniORB-4.2.1/lib/i586_lin
ux_2.0
root@debian:~/omniORB-4.2.1/src/examples/echo# export LD_LIBRARY_PATH
```

Nun kann das C++ Example erfolgreich ausgeführt werden.

```
root@debian:~/omniORB-4.2.1/src/examples/echo# ./eg1
I said, "Hello!".
The Echo object replied, "Hello!".
Segmentation fault
```

Das ist die Referenz von dem Objekt.

```
root@debian:~/omniORB-4.2.1/src/examples/echo# ./eg3_impl
IOR:010000000d00000049444c3a4563686f3a312e3000000000010000000000000068000000010102000f000000313
9322e3136382e39322e31353200001cad00000e000000fefb24fc5600002c5d0000000000000000200000000000000008
0000000100000000545441010000001c0000000100000001000100010000000100010509010100010000000009010100
```

Das C++ Examples eq3 wird ausgeführt und die Funktionalität wird somit getestet. Das impl ist der Server und clt ist der Client.

```
root@debian:~/omniORB-4.2.1/src/examples/echo# ./eg3_clt
Caught NO_RESOURCES exception. You must configure omniORB with the location
of the naming service.
hello: The object reference is nil!

hello: The object reference is nil!

hello: The object reference is nil!

hello: The object reference is nil!

hello: The object reference is nil!

hello: The object reference is nil!

hello: The object reference is nil!

hello: The object reference is nil!

hello: The object reference is nil!

hello: The object reference is nil!
```

jacORB:

Das Paket jacORB wird runtergeladen.

```
root@debian:~# wget http://www.jacorb.org/releases/3.7/jacorb-3.7-binary.zip
--2016-03-30 12:31:57--  http://www.jacorb.org/releases/3.7/jacorb-3.7-binary.zi
p
Resolving www.jacorb.org (www.jacorb.org)... 80.68.89.40
Connecting to www.jacorb.org (www.jacorb.org)|80.68.89.40|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9585148 (9.1M) [application/zip]
Saving to: 'jacorb-3.7-binary.zip.1'
```

Diese Datei wird nun entpackt.

```
root@debian:~# unzip jacorb-3.7-binary.zip
Archive:  jacorb-3.7-binary.zip
   creating: jacorb-3.7/
   creating: jacorb-3.7/demo/
   creating: jacorb-3.7/demo/dii/
   creating: jacorb-3.7/demo/dii/src/
   creating: jacorb-3.7/demo/dii/src/main/
   creating: jacorb-3.7/demo/dii/src/main/java/
   creating: jacorb-3.7/demo/dii/src/main/java/org/
   creating: jacorb-3.7/demo/dii/src/main/java/org/jacorb/
   creating: jacorb-3.7/demo/dii/src/main/java/org/jacorb/demo/
```

Es wird vorausgesetzt, dass maven bereits schon installiert wurde.

```
root@debian:~# apt-get install maven
Reading package lists... Done
Building dependency tree
Reading state information... Done
maven is already the newest version.
The following packages were automatically installed and are no longer required:
  python-defusedxml python-soappy python-wstools
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 233 not upgraded.
```

Das Befehl mvn clean löscht Datei, die während Builden erzeugt wurden.

```
root@debian:~/jacorb-3.7/demo/hello# mvn clean
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building HelloDemo 3.7
[INFO] ------------------------------------------------------------------------
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven
-clean-plugin/2.5/maven-clean-plugin-2.5.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-
clean-plugin/2.5/maven-clean-plugin-2.5.pom (4 KB at 3.3 KB/sec)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven
-clean-plugin/2.5/maven-clean-plugin-2.5.jar
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-
clean-plugin/2.5/maven-clean-plugin-2.5.jar (25 KB at 84.4 KB/sec)
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ hello ---
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 5.521 s
[INFO] Finished at: 2016-03-31T21:14:45+02:00
[INFO] Final Memory: 10M/25M
```

Mvn install durchführen

```
root@debian:~/jacorb-3.7/demo/hello# mvn install
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building HelloDemo 3.7
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-enforcer-plugin:1.0:enforce (enforce-maven) @ hello ---
[INFO]
[INFO] --- buildnumber-maven-plugin:1.3:create (buildnumber) @ hello ---
[INFO] ShortRevision tag detected. The value is '8'.
[INFO] Executing: /bin/sh -c cd /root/jacorb-3.7/demo/hello && git rev-parse --v
erify --short=8 HEAD
[INFO] Working directory: /root/jacorb-3.7/demo/hello
[INFO] Storing buildNumber: null at timestamp: 31 March 2016
[WARNING] Cannot get the branch information from the git repository:
Detecting the current branch failed: /bin/sh: 1: git: not found

[INFO] ShortRevision tag detected. The value is '8'.
[INFO] Executing: /bin/sh -c cd /root/jacorb-3.7/demo/hello && git rev-parse --v
erify --short=8 HEAD
[INFO] Working directory: /root/jacorb-3.7/demo/hello
[INFO] Storing buildScmBranch: UNKNOWN_BRANCH
[INFO]
[INFO] --- buildnumber-maven-plugin:1.3:create-timestamp (year) @ hello ---
```

Aufwandschätzung:

Geschätzter Aufwand:

In der Aufwandschätzung werden auch die Überlegung und die Planung berücksichtigt und nicht nur die Arbeitsdurchführung. Dabei wird angegeben, wie viel Zeit für die Erstellung des Programmes und der Dokumentation benötigt wird. Der Aufwand wird in Zeit gemessen bzw. die Dauer wird in Stunden und Minuten angegeben. Die folgende Tabelle zeigt auch die einzelnen Aufgabenteile für die Realisierung.

Teile der Aufgabe	Dauer (h)	GD (h)
omniorb unter Debian oder Linux zum Laufen bringen	2	10
jacorb unter Debian oder Linux zum Laufen bringen	2	
Verwendung und Ausführung von Examples (Java, C++)	1	
Protokoll schreiben	1,5	
Source Code - Dokumentation (Java Doc) + Erstellung von Code Snippets,	0,5	
JUNIT Testing (Testen aller Methoden)	2	
Verwendung des IDL - Interfaces	0,5	
Recherchieren im Internet für die wichtigsten Tools (jacorb, omniorb)	0,5	

Realer Aufwand:

In diesem Unterpunkt wird erklärt, wie groß der Aufwand in Stunden in der Realität für uns, wie viel Zeit wir gebraucht haben, alle notwendigen Arbeiten durchzuführen. Die Idee dahinter ist, dass die Anzahl der Stunden für die einzelnen Teile der Aufgabe variieren kann, d.h die Gesamtdauer kann entweder höher oder niedriger werden oder im idealsten Fall gleich (fast gleich) wie bei dem geschätzten Aufwand bleiben. Damit möchte man den Soll-Wert mit dem Ist-Wert in Stunden vergleichen.

Teile der Aufgabe	Dauer (h)	GD (h)
omniorb unter Debian oder Linux zum Laufen bringen	2	12
jacorb unter Debian oder Linux zum Laufen bringen	2,5	
Verwendung und Ausführung von Examples (Java, C++)	1,5	
Protokoll schreiben	1,5	
Source Code - Dokumentation (Java Doc) + Erstellung von Code Snippets,	0,5	
JUNIT Testing (Testen aller Methoden)	2	
Verwendung des IDL - Interfaces	1,5	
Recherchieren im Internet für die wichtigsten Tools (jacorb, omniorb)	0,5	

Github Repository:

https://github.com/bobanjevtic-tgm/Verteilte-Objekte-mit-CORBA

Quellen:

- http://omniorb.sourceforge.net/, zuletzt aufgerufen am 30.03.2016
- http://www.jacorb.org/download.html, zuletzt aufgerufen am 30.03.2016
- http://omniorb.sourceforge.net/omni42/omniORB.pdf, zuletzt aufgerufen am 01.04.2016