A. Benutzungsanleitung

Die in dieser Arbeit vorgestellten Implementierungen stehen unter der GNU-Public-License [Fou91] und sind damit frei verfügbar. Die folgenden Kapitel beschreiben die Installation, Konfiguration und die allgemeine Bedienung des Systems. Für weitere Details sei auf die entsprechenden Kapitel in dieser Arbeit verwiesen.

A.1. Installation

Für die Installation des Systems ist zunächst die aktuelle Distribution unter der Adresse

```
http://www.inf.fu-berlin.de/~tnfink/ECL/Dist/ecl.jar
```

zu laden. Die Distribution ist dann in dem gewünschten Verzeichnis `INSTDIR` zu entpacken:

```
cd INSTDIR
jar xf ecl.jar
```

Nun ist die allgemeine Konfigurationsdatei `ECL/metacomprc.template` in das Heimatverzeichnis des Benutzers unter dem Namen `.metacomprc` zu kopieren. Die Einträge in der Datei sind geeignet anzupassen. Eine Beschreibung der Einträge ist dem Anhang A.4 zu entnehmen. Zusätzlich ist die Konfigurationsdatei `.jacorb_properties` für das eingesetzte CORBA-System anzupassen. Eine Beschreibung dieser Datei ist [jac02] zu entnehmen.

ECL benutzt einige externe Bibliotheken, die in Abbildung A.1 aufgeführt sind. Um die Installation zu vereinfachen, sind alle benötigten Bibliotheken in der ECL-Distribution enthalten. Die einzigen Ausnahmen sind die IAIK-Bibliotheken, die nicht frei zu Verfügung stehen. Diese werden für sicherheitsrelevante Funktionalität, wie Authentisierung, Vertraulichkeit etc., in der Amica-Infrastruktur benutzt. Wird diese Funktionalität benötigt, sind die beiden Bibliotheken `iaik_jce_full.jar` und `iaik_ssl.jar` über die in der Abbildung A.1 enthaltene Kontaktadresse zu besorgen und in das Verzeichnis `ECL/lib/` zu kopieren.

Die ausführbaren Programme bestehen im Wesentlichen aus einem Java-Archiv mit einer Startklasse. Um die Handhabung der Programme zu erleichtern, befinden sich in dem Verzeichnis `ECL/bin/` Skripte, welche die Archive zusammen mit den benötigten Bibliotheken in den Klassenpfad einer Java-Umgebung legen und über die Startklasse das Programm starten. Diese Skripte sind für den Bourne-Shell-Kommandozeileninterpreter geschrieben,

Acme/Armani
Acme und Armani sind Architekturbeschreibungssprachen. Die eingesetzte Bibliothek unterstützt die Übersetzung von Architekturen aus einer textuellen Repräsentation in eine interne Darstellung und deren Rückübersetzung. Die unter dem unteren Verweis verfügbare Version kann beliebig benutzt werden, solange die mitgelieferte Urheberrecht-Notiz weitergegeben wird. Die bei der Implementierung eingesetzte Bibliothek beruht aber auf einer aktuelleren Version, die direkt von dem Autor stammt und zusätzlich leicht modifiziert wurde.
`http://www-2.cs.cmu.edu/~able/software.html`

Graph Editing Framework (GEF)
GEF ist eine Java Bibliothek zur grafischen Bearbeitung von Graphen. Sie unterliegt der BSD-Lizenz [Ini02].
`http://gef.tigris.org/`

JacORB
JacORB ist eine Implementierung des CORBA-Standards für Java. Sie unterliegt der GNU-General-Public License (GPL) [Fou91].
`http://www.jacorb.org/`

IAIK-JCE
Dies ist eine Implementierung der Java-Cryptography-Extension von SUN, einer Sammlung von Schnittstellen für kryptographische Aktionen. Sie ist kommerziell und damit nicht frei verfügbar. Die Implementierungen wurden mit Evaluationsversionen durchgeführt.
`http://jce.iaik.tugraz.at/`

JUnit
JUnit stellt eine Umgebung für Regressionstestläufe zur Verfügung. Die Bibliothek steht unter der Common-Public-License von IBM zur Verfügung [IBM].
`http://www.junit.org/index.htm`

Oscar
Oscar ist eine Implementierung des OSGi-Standards [osg00] und unterliegt der GPL [Fou91].
`http://oscar-osgi.sourceforge.net/`

Abbildung A.1.: Auflistung der eingesetzten externen Bibliotheken

der auf allen gängigen Unix-Systemen verfügbar ist [Her99]. Für andere Systeme, wie z.B. Windows 2000, müssen diese Skripte zuerst portiert werden. Für die Ausführung wird ein Java-Laufzeitsystem der Version 1.2 oder höher benötigt.

Der Quellcode der Implementierungen steht als JAR-Archiv in der Datei `ECL/src.jar` zur Verfügung. Für die Übersetzung werden neben den oben erwähnten IAIK-Bibliotheken ein Java-Entwicklungspaket der Version 1.2 oder höher und das Werkzeug Ant [AJP02] benötigt.

A.2. Starten und Benutzung des grafischen ECL-Editors

Um den grafischen Editor des ECL-Systems zur Erstellung von Applikationen zu aktivieren, ist das Skript `ECL/Bin/startECLEditor.sh` zu starten. Darauf erscheint ein Fenster mit einer Menüleiste. Neue Applikationen können mit dem Menüeintrag `File->New` erzeugt werden. Vorhandene Applikationen können über `File->Load` geladen werden.

In dem Editor können mehrere Applikationen gleichzeitig bearbeitet werden. Jede Applikation wird in einem eigenen Unterfenster dargestellt und bearbeitet. Abbildung A.2 zeigt, wie zwei Applikationen gleichzeitig editiert werden. Die Erstellung von Applikationen wird in Kapitel 5.2.1 genauer beschrieben.

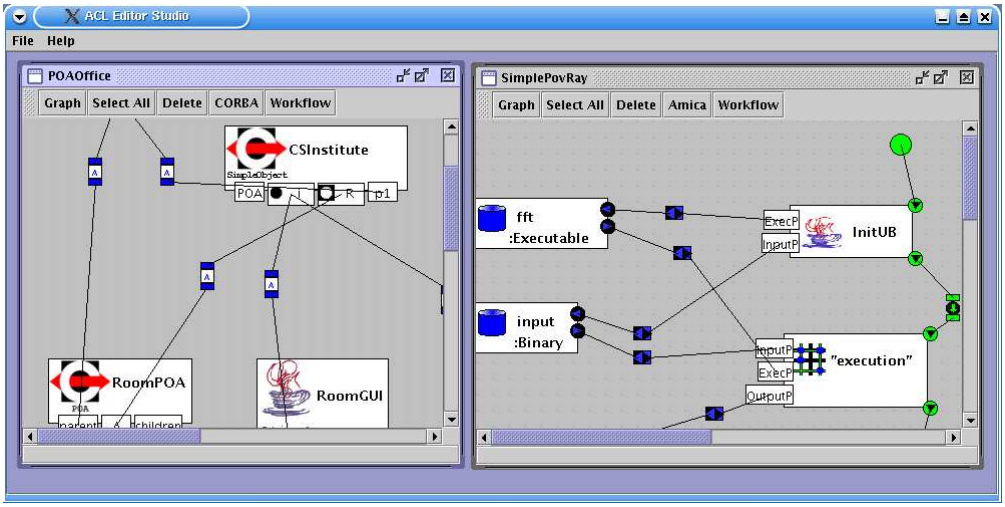

Abbildung A.2.: Bearbeitung von zwei Applikationen im ECL-Editor

Applikationen werden über den Menüpunkt `Graph->Save` in ihrem Unterfenster gespeichert. Mit dem Menüpunkt `Graph->Export` lässt sich die Applikationsarchitektur in unterschiedlichen Bildformaten speichern. Mit `Graph->Print` kann man sie ausdrucken. Die Übersetzung einer Applikation geschieht über `Graph->Compile`. Nach der Übersetzung wird der Applikationsentwickler gefragt, in welcher Datei er das Applikationskompilat abspeichern möchte.

A.3. Starten von Applikationen

Bevor Applikationen gestartet werden können, ist zuerst die Infrastruktur aufzusetzen, die von den eingesetzten Erweiterungsmodulen benutzt wird. Wie Applikationen gestartet werden, hängt wiederum von den eingesetzten Infrastrukturen ab. Im folgenden wird die Vorgehensweise für das Amica-System und für die Konfiguration CORBA-basierter Applikationen beschrieben.

A.3.1. Aufsetzen von Amica-Applikationen

Die einzelnen Elemente des Amica-Systems sind in den Kapiteln 8 und 9 ausführlich beschrieben worden. Um auf einem Rechner die einzelnen Server zu starten, wurde eine Starterapplikation mit einer grafischen Oberfläche implementiert. Neben einer Vereinfachung des Aufsetzens der Infrastruktur ermöglicht sie auf einfache Weise, mehrere Server in einer Java-Umgebung zu platzieren, um Speicher- und Rechenressourcen zu sparen. Sie wird mit dem Skript `ECL/Bin/startAmicaInfrastructure.sh` aktiviert.

Abbildung A.3 zeigt die grafische Oberfläche der Applikation. Sie ist unterteilt in drei Teile, die über Karteireiter ausgewählt werden. Mit der über `CORBA Basics` erreichbaren Schnittstelle lassen sich die einzelnen CORBA-Dienste starten. In einem funktionsfähigen Amica System müssen alle Dienste aktiviert werden.

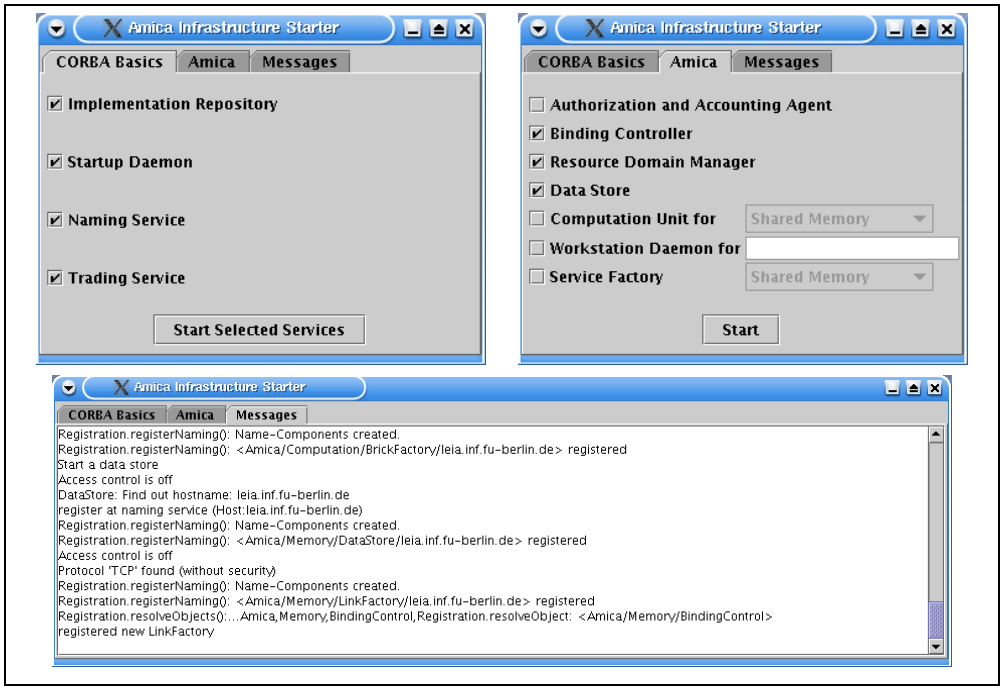

Abbildung A.3.: Grafische Applikation zum Aufsetzen von Amica

Über den Karteireiter `Amica` lassen sich die speziellen Komponenten von Amica starten. Der `ResourceDomainManager` dient für die Verwaltung einer Amica Domäne und muss gestartet werden. Zusätzlich muss ein `Binding`

`Controller` gestartet werden. Alle anderen Komponenten sind optional.

Es werden zwei Arten von Recheneinheiten (`Computation Unit`) und Rechendienstfabriken (`Service Factory`) unterstützt. Eine Art unterstützt Multiprozessorsysteme mit gemeinsamem Speicher, die andere unterstützt Multiprozessorsysteme mit verteiltem Speicher, wie z.B. Workstation Cluster. Welche Art gestartet wird, wird mit dem Knopf jeweils rechts von dem Auswahlknopf bestimmt.

Neben der Recheneinheit ist für Systeme mit verteiltem Speicher auch auf jedem Rechner zusätzlich ein Hintergrundprozess zu starten, der von der Recheneinheit benutzt wird. Damit dieser Hintergrundprozess seine Recheneinheit lokalisieren kann, benötigt er den IP-Namen des Rechners, auf dem die Recheneinheit läuft. Dieser wird in dem Textfeld neben dem zugehörigen Auswahlknopf eingegeben.

Die letzte Oberfläche, die über den Karteireiter `Messages` erreichbar ist, besteht nur aus einem Textfeld. Dieses gibt alle Nachrichten wieder, die beim Hochfahren und beim Betrieb der einzelnen Elemente erzeugt werden.

Nachdem nun das Amica-System hochgefahren wurde, können Applikationen gestartet werden. Auch hierfür gibt es eine spezielle Applikation mit grafischer Oberfläche, die mit dem Skript `Bin/startAmicaApplication.sh` gestartet wird. Abbildung A.4 zeigt ihre Oberfläche. In dem oberen Textfeld ist die Datei mit dem Applikationskompilat einzugeben. Dann kann die Applikation mit dem `Start`-Knopf gestartet werden. Die Ausgaben der laufenden Applikation sind in dem mittleren Textfeld zu beobachten.

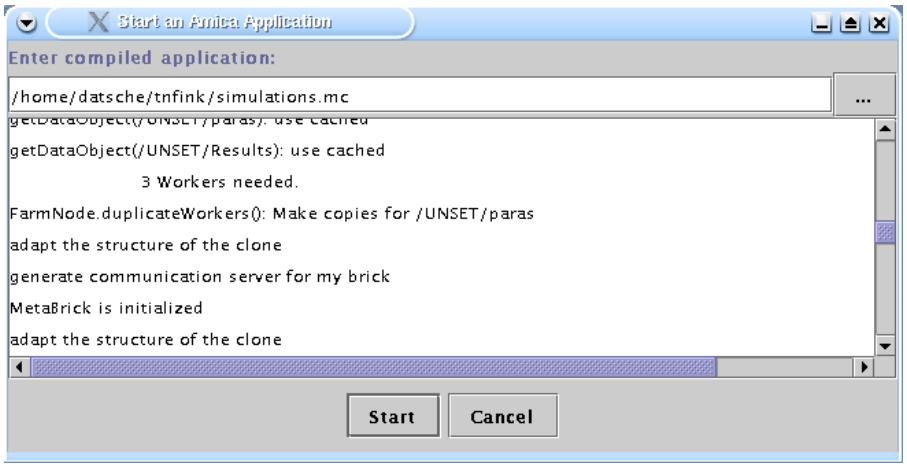

Abbildung A.4.: Starten einer Amica-Applikation

A.3.2. Konfiguration von CORBA-basierten Applikationen

Bevor die Infrastruktur für CORBA-basierte Applikationen aufgesetzt werden kann, ist die Konfigurationsdatei `.metacomprc` an das lokale System anzupassen. Die Konfigurationseinträge sind in Kapitel A.4 detailliert beschrieben. Dann kann mit dem Skript `Bin/startCORBA_Box.sh` auf den

gewünschten Rechnern ein Knoten der Laufzeitumgebung gestartet werden. Jeder Knoten unterstützt eine Webschnittstelle, die über den Port `8080` mit einem beliebigem Webbrowser angesteuert werden kann. Es ist zu beachten, dass die CORBA-Infrastruktur unter einer Java-Laufzeitumgebung der Version 1.4 nicht fehlerfrei funktioniert. Bei Einsatz der Versionen 1.2 oder 1.3 ergeben sich keine Probleme.

Unter der URL `http://host:8080/ace/index.html` ist die Oberfläche für das Aufsetzen einer Applikation abgelegt. Abbildung A.5 zeigt eine Abbildung der Oberfläche. In dem Feld `Application URL` ist eine URL auf die übersetzte Applikationsarchitektur einzugeben. Jeder Knoten der Laufzeitumgebung, der von der Applikation eingesetzt wird, muss einen lesenden Zugriff auf die von dieser URL referenzierte Applikationsarchitektur besitzen. In dem Feld `Start component` ist der Name der Applikationskomponente einzugeben, die gestartet werden soll. Wird das Feld leer gelassen, werden alle Komponenten gestartet.

Abbildung A.5.: Schnittstelle zum Aufsetzen einer CORBA-Applikation

Über den Verweis `CORBA Monitor` auf der Oberfläche für das Starten einer Applikation gelangt man zu einer Oberfläche, mit der sich der aktuelle Status der CORBA-Infrastruktur beobachten und manipulieren lässt. Abbildung A.6 zeigt diese Oberfläche. Es werden alle laufenden Applikationen mit all ihren aktiven Komponenten, bzw. erzeugten Laufzeitexemplaren, dargestellt. In dem abgebildeten System läuft nur die Applikation `HomeSystem`, die ausschließlich den Rechner `leia` benutzt. Sie besitzt sechs Laufzeitexemplare.

Über die Oberfläche ist es zusätzlich möglich, mit dem Knopf `Shutdown` bestimmte Laufzeitexemplare zu terminieren. Dabei werden rekursiv auch alle Exemplare terminiert, welche dieses Exemplar benötigen. Der genaue Algorithmus ist in Kapitel 11.2.6 beschrieben. Schließlich gelangt man über den Verweis `CORBA-Log` zu einer Oberfläche, welche die angefallenen Systemnachrichten anzeigt. Abbildung A.7 zeigt ein Bildschirmfoto.

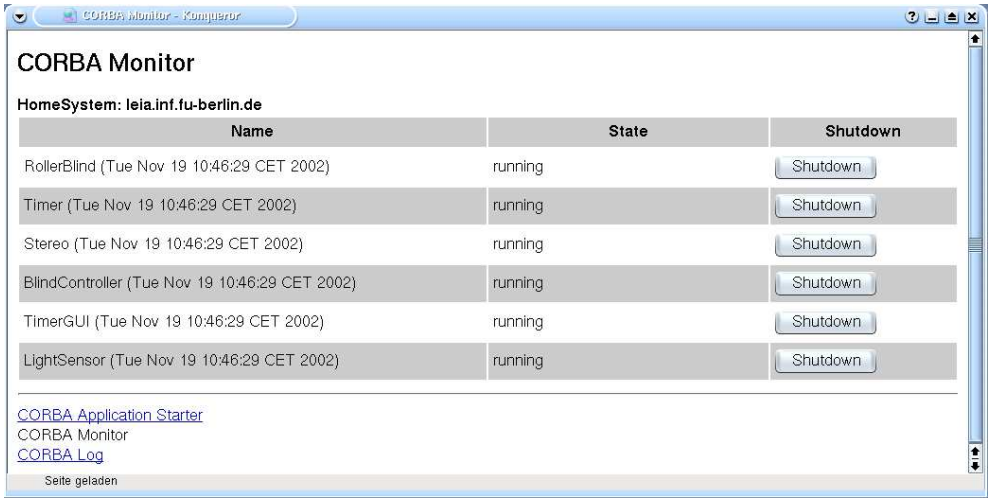

Abbildung A.6.: Betrachtung des Zustands der CORBA-Infrastruktur und Terminierung einzelner Laufzeitexemplare

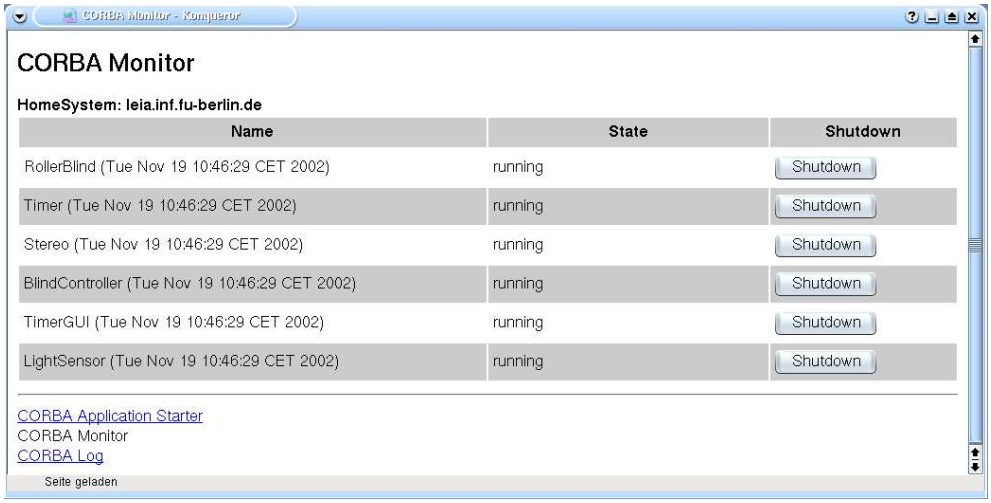

Abbildung A.7.: Systemnachrichten der CORBA-Infrastruktur

A.4. ECL-Konfigurationsdatei

Die ECL-Konfigurationsdatei befindet sich unter dem Namen `.metacomprc` in dem Heimatverzeichnis des Benutzers. Als Heimatverzeichnis wird der Wert der Java-Systemvariablen `user.home` benutzt. Sie besteht aus beliebig vielen zeilenweisen Einträgen, die folgender Syntax unterliegen:

```
<host>/<name>=<value>
```

Der Wert von `<host>` ist eine IP-Adresse, die bestimmt, für welchen Rechner dieser Eintrag gültig ist. Wird statt einer Adresse der Wert * benutzt, gilt der Eintrag für alle Rechner. Der Wert `<name>` spezifiziert den Namen einer Eigenschaft und `<value>` dessen Wert. In dem restlichen Kapitel werden die unterstützten Eigenschaften beschrieben.

Computation.workingDir=DIR bestimmt das Arbeitsverzeichnis, welches von Recheneinheiten benutzt wird, falls Rechendienste ein Arbeitsverzeichnis benötigen.

CORBA.orbArgs={STR,STR}* spezifiziert Parameter, welche bei der Erzeugung von neuen CORBA-ORBs diesen übergeben werden. Dies gilt nur für ORBs, die von der Infrastruktur für CORBA-basierte Applikationen erzeugt werden (s. Kapitel 12).

CORBA.startNS=FILE verweist auf ein Programm, welches einen CORBA-Namensdienst startet. Es wird von der Infrastruktur für CORBA-basierte Applikationen eingesetzt (s. Kapitel 12).

ctrlPollIntervall=INT bestimmt den Zeitabstand in *ms* zwischen zwei Überprüfungen einer Verbindung zwischen zwei Datenspeichern. Bei dieser Überprüfung wird verifiziert, ob die mittlere Zeit zwischen zwei Übertragungen den Wert in `meanTimeOut` überschritten hat.

dynamicProviders=STR{,STR}* enthält Namen von Klassen, die Implementierungen der Java-Cryptography-Extension bereitstellen. Diese können dann bei der verschlüsselten Übertragung von Datenobjekten eingesetzt werden. Für weitere Details siehe [Kra99].

generalBF.transferBricksToCu=BOOL bestimmt, ob das Objekt, das einen Rechendienst bereit stellt, vor seiner Ausführung in den Prozess der Recheneinheit übertragen wird (s. Kapitel 9.2.2).

MBNWaitForTrader=INT bestimmt die maximale Wartzeit in *ms*, die nach einer Suchanfrage bzgl. einer Dienstfabrik auf die Antwort von dem Vermittlungsdienst gewartet wird (s. Kapitel 9.2.2).

meanTimeOut=INT bestimmt die maximale mittlere Zeit in *ms*, die zwischen zwei Übertragungen zwischen zwei Datenspeichern vergehen darf, bevor die Verbindung geschlossen wird, d.h. die entsprechenden Verbindungsobjekte gelöscht werden (s. Kapitel 8.1.2).

namingURL=URL enthält eine Referenz auf eine Datei, die eine CORBA-Referenz (IOR) auf den Namensdienst enthält.

protocolFactories=STR{,STR}* enthält die Namen der Protokollfabrikklassen, die von Verbindungsobjekten benutzt werden können (s. Kapitel 8.1.2).

starter.domainName=STR bestimmt den Namen der Amica-Domäne, die mit dem Amica-Startprogramm erzeugt wurde.

starter.imrfile=FILE bestimmt die Datei, in welcher das Implementation-Repository des CORBA-Systems seine Daten ablegt, sofern es mit dem Amica-Startprogramm erzeugt wurde.

starter.namingfile=FILE bestimmt die Datei, in welcher das Amica-Startprogramm für die Amica-Infrastruktur die CORBA-Referenz auf den Namensdienst ablegt (s. Kapitel A.3.1).

startExec=FILE bestimmt das Programm, das von dem allgemeinen Rechendienst zur Ausführung beliebiger Programme benutzt wird, um andere Programme zu starten (s. Kapitel 10.2).

starter.tradingDB=DIR bestimmt das Verzeichnis, in dem der Vermittlungsdienst seine Daten ablegt, wenn er von dem Amica-Startprogramm erzeugt wurde (s. Kapitel A.3.1).

starter.tradingIOR bestimmt die Datei, in welcher das Amica-Startprogramm für die Amica-Infrastruktur die CORBA-Referenz auf den Vermittlungsdienst ablegt (s. Kapitel A.3.1).

storeDataBase=DIR bestimmt das Verzeichnis, in dem ein Datenspeicher seine Datenobjekte ablegen kann. Dieses wird zur Zeit allerdings noch nicht unterstützt.

tradingURL=URL enthält eine Referenz auf eine Datei, die eine CORBA-Referenz (IOR) auf den Vermittlungsdienst enthält.

trustedKM=URL enthält eine Referenz in Form einer URL im IIOP-Format auf den Schlüsselverwalter, der für verschlüsselte Übertragungen von Datenobjekten eingesetzt wird.

WaitNSRegisterPollTime enthält die Wartezeit in *ms* zwischen zwei Anfragen beim Namensdienst in der Hilfsfunktion

```
amica.util.WaitNSRegister.waitForNSRegister().
```

Für weitere Beschreibung siehe die Klassendokumentation.

WSCDaemon.updateLoadIntv=INT bestimmt die Wartezeit zwischen zwei Messungen der CPU-Auslastung. Dieser Wert wird von den Programmen benutzt, welche in einem Workstationcluster die einzelnen Workstations verwalten.