Service Discovery in CORBA

Seminararbeit

vorgelegt am

Lehrstuhl für Praktische Informatik IV
Prof. Dr. Effelsberg
Universität Mannheim

im

Januar 2001

von

Markus Aleksy

Inhaltsverzeichnis

1. Einleitung

Die Object Management Group (OMG) ist ein Zusammenschluß von Hardwareherstellern, Softwareentwicklern, Netzwerkbetreibern und Nutzern von Software. Sie wurde 1989 von acht Firmen gegründet (3Com, American Airlines, Canon, Data General, Hewlett-Packard, Philips Telecommunications, Sun Microsystems und Unisys), öffnete sich aber nach wenigen Monaten als unabhängige Organisation für andere Unternehmen. Momentan hat die OMG über 800 Mitglieder. Kern der Arbeit der OMG ist die Spezifikation einer Architektur für die Verteilung und Kooperation objektorientierter Softwarebausteine in heterogenen, vernetzten Systemen. Die OMG setzt sich für die modulare Softwareentwicklung ein, also eine Unterteilung der Anwendung in Komponenten, d.h., eine Menge von Objekten, die gemeinsam eine bestimmte Aufgabe erfüllen. Im Gegensatz zu anderen Standardisierungsorganisationen (beispielsweise OSF, Open Software Foundation) produziert die OMG jedoch selber keine Software. Es werden vielmehr Standards (Richtlinien, Spezifikationen, Architekturmodelle) erarbeitet mit grundlegenden Anforderungen an Anwendungen und deren Komponenten, die von allen standardkonformen Softwareprodukten eingehalten werden müssen. Das Einhalten der Vorgaben soll es zum einen ermöglichen, daß Anwendungen ohne Änderung am Programmcode auf verschiedenen Hardwarearchitekturen und Betriebssystemen laufen. Zum anderen sollen getrennt voneinander entwickelte, in verschiedenen Programmiersprachen geschriebene und auf verschiedenen Rechnersystemen an verschiedenen Orten laufende Programme miteinander Nachrichten austauschen können.

Im *Object Management Architecture-Guide* (OMA), dem ersten wichtigen Dokument der OMG, werden Ziele und Vorgehensweise der OMG sowie die grundlegende Objektverwaltungsarchitektur dargelegt. Auf dieser Basis werden allgemeine Anforderungen an das Objektmodell aufgestellt, die jedes System erfüllen muß, wenn es dem Standard entsprechen will: das „*Core Object Model*“. Ein zentraler Aspekt der OMA ist die Beschreibung von Schnittstellen („Interfaces“), d.h. den äußerlich „sichtbaren“ Eigenschaften von Objekten. Die OMG stellt dafür eine eigene *Interface Definition Language* (IDL) bereit, die diese Eigenschaften in einer standardisierten und implementierungsunabhängigen Weise beschreibt. Um ein Objekt zu implementieren, kann auf verschiedene Programmiersprachen zurückgegriffen werden (z.B. C++, Smalltalk, aber auch nicht-objektorientierte Sprachen wie Cobol). Für diese Sprachen definiert die OMG *Language Mappings*, die angeben, wie die in IDL angegebenen Interfaces und Datentypen in Konstrukte der Programmiersprache

umgesetzt werden. Den Kern der Object Management Architecture bildet der *Object Request Broker* (ORB). Er ist die Kommunikationskomponente, die für das Auffinden und die Initialisierung von Objekten, sowie für die Herstellung der Kommunikation zwischen Client und Server verantwortlich ist. Er wird von Objekten zur Kommunikation mit anderen Objekten in einer verteilten Umgebung genutzt und ermöglicht als „Postverteiler“ das Weiterleiten eines Aufrufs zu einem Zielrechner und dort zu dem betreffenden Objekt. Weiterhin werden vier Kategorien von Anwendungskomponenten beschrieben, unterschieden danach, wie sehr die bereitgestellten Dienste auf ein bestimmtes Anwendungsprofil zugeschnitten sind:

- **Object Services:** Hierbei handelt es sich um generelle, systemnahe Erweiterungen der Grundfunktionalität eines ORBs, welche für die grundsätzliche Funktionsweise einer verteilten Anwendung von Bedeutung sind. Ein Anwendungsprogrammierer kann bei Bedarf auf diese Dienste zurückgreifen wenn seine Anwendung zusätzliche Funktionalität wie z.B. Transaktionen, Sicherheit oder Persistenz benötigt.
- **Common Facilities:** Hiermit sind komplexere, anwendungsnahe Funktionen, die wesentliche Teile einer Anwendung ausmachen, angesprochen Dazu zählen z.B. Funktionen für Benutzeroberflächen, für die Dokumentenverwaltung oder Druckdienste, d.h., sie sind im Vergleich zu den zuvor genannten Object Services auf einer höheren Anwendungsebene angesiedelt.
- **Domain Interfaces:** Sie sind Teillösungen für bestimmte Anwendungsgebiete bzw. Branchen, die in bezug auf ihren prinzipiellen Aufbau standardisiert werden. Dazu gehören z.B. Finanz- und Telekommunikationsschnittstellen.
- **Application Objects:** Diese sind nicht von der OMG standardisierte konkrete Produkte, also die eigentlichen Anwendungen. Sie entsprechen Anwendungsprogrammen, wie z.B. Textverarbeitungs- oder CAD-Programmen.

2. Interoperabilität

Der ORB selbst schafft als Kommunikationsmedium bereits die Grundvoraussetzungen für die Interoperabilität zwischen Anwendungen – unabhängig von den eingesetzten Programmiersprachen und Plattformen. Sobald jedoch ORBs verschiedener Hersteller verwendet werden, sind weitergehende Konzepte nötig. Die von der OMG definierte Interoperabilität zwischen verschiedenen ORBs basiert auf zwei Elementen:

- Definition eines Kommunikationsprotokolls sowie
- Einführung einer eindeutigen Objektreferenzierung.

Eine praktische Untersuchung der Interoperabilität verschiedener CORBA-ORBs erfolgte u.a. in [6] (siehe auch `http://www.wifo.uni-mannheim.de/IIOP`).

2.1. Die von CORBA definierten Protokolle

Eine der wichtigsten Neuerungen des Standards CORBA 2.0 gegenüber der Version 1.1 war die Definition eines Protokolls zur Kommunikation zwischen den ORBs verschiedener Hersteller. Das *General Inter-ORB Protocol* (GIOP) spezifiziert eine standardisierte Übertragungssyntax - die *Common Data Representation* (CDR) - sowie einige Nachrichtenformate. Zu den Merkmalen der CDR gehört eine komplette Anbindung aller in der *Interface Definition Language* (IDL) definierten Datentypen und die Unterstützung unterschiedlicher Byte-Anordungen. Die GIOP-Definition enthält folgende Nachrichtenformate:

- Request:
  Dieses Format ermöglicht dem Client die Zugriffe auf die Objektimplementierung.
- Reply:
  Mit Hilfe dieses Formates werden die Ausgabe- und/oder Rückgabewerte des Servers an den Client zurückgeliefert. Falls die Objektimplementierung ihren Aufenthaltsort geändert hat, enthält die Antwort ein LocationForward und die neue Objektreferenz.
- CancelRequest:
  Dieses Format dient dazu dem Server mitzuteilen, daß der Client auf die Beantwortung seiner Anfrage verzichtet.

- LocateRequest:

  Mittels LocateRequest kann der Client überprüfen, ob eine Objektreferenz gültig ist.
- LocateReply:

  Mit Hilfe dieses Nachrichtenformates beantwortet der Server eine LocateRequest-Anfrage. Die Antwort enthält eine Nachricht, ob der Server das Objekt kennt oder nicht. Wenn das Objekt seinen Aufenthaltsort geändert hat und der Server den neuen ORB kennt, schickt er die aktuelle Objektreferenz als Antwort.
- CloseConnection:

  Soll eine Verbindung vorzeitig beendet werden (z.B. wenn der Server noch nicht alle Anfragen beantwortet hat), so muß dieses Format dazu benutzt werden.
- MessageError:

  Dieses Format kann sowohl vom Client, als auch vom Server benutzt werden und dient dazu, Kommunikationsprobleme anzuzeigen.

Die Einführung des CORBA-Standards 2.1 erweiterte die GIOP-Nachrichtenformate um die Fragment-Nachricht. Durch ihren Einsatz lassen sich Anfragen und Antworten in kleinere Pakete (Fragmente) aufteilen. Diese beinhalten eine Kennung, ob eine weitere Fragment-Nachricht zu erwarten ist oder ob es das letzte Paket war. Die Anfrage bzw. die Antwort wird erst ausgewertet, nachdem das letzte Paket angekommen ist. Daneben wurden einige von den ORBs intern genutzte Datenstrukturen, verändert. Diese betreffen jedoch nicht den Anwendungsprogrammierer und sind nur für die Entwickler von ORBs und (Halb)Brücken von Interesse. Tabelle 1 enthält einen Überblick über die verschiedenen Nachrichtenformate.

Art der Nachricht	Urheber	Wert	Definiert in GIOP-Version
`Request`	Client	0	1.0, 1.1, 1.2
`Reply`	Server	1	1.0, 1.1, 1.2
`CancelRequest`	Client	2	1.0, 1.1, 1.2
`LocateRequest`	Client	3	1.0, 1.1, 1.2
`LocateReply`	Server	4	1.0, 1.1, 1.2
`CloseConnection`	Server	5	1.0, 1.1, 1.2
`MessageError`	Beide	6	1.0, 1.1, 1.2
`Fragment`	Beide	7	1.1, 1.2

Tabelle 1: Nachrichtenformate des GIOP-Protokolls

Beim GIOP handelt es sich um ein abstraktes Protokoll. Die tatsächliche Kommunikation findet beispielsweise über das *Internet Inter-ORB Protocol* (IIOP) statt. Im IIOP ist spezifiziert, wie GIOP-Nachrichten über TCP/IP-Verbindungen ausgetauscht werden können.

Damit die Interoperabilität zwischen ORBs mit unterschiedlichen Protokollversionen nicht gefährdet ist, werden von der OMG Angaben dazu gemacht, wann welches Protokoll unterstützt werden muß. So kann ein IIOP 1.1 Client entweder beide GIOP-Versionen (1.0 und 1.1) oder nur die Version 1.1 unterstützen. Ein Server, der die IIOP-Version 1.1 verwendet, muß in der Lage sein, Anfragen in beiden Formaten, d.h. GIOP 1.0 und 1.1 entgegenzunehmen. Ein IIOP 1.2 Client muß die GIOP-Version 1.2 unterstützen und kann zusätzlich dazu auch die GIOP-Vorgängerversionen 1.0 und 1.1 unterstützen. Ein IIOP 1.2 Server muß in der Lage sein, alle Nachrichten aller vorhandenen GIOP-Versionen zu beantworten. Die Antwort, die er zurückschickt, entspricht der Protokollversion, in der die Anfrage formuliert war.

Neben dem IIOP können aber auch weitere, sogenannte *Environment-Specific-Inter-ORB-Protocols* (ESIOPs) verwendet werden.

2.2. Die Inter-ORB-Referenz

Die Einführung der *Inter-ORB Reference (IOR)* soll die weltweit eindeutige Referenzierung von Objekten ermöglichen. Bei der IOR handelt es sich um eine spezielle Objektreferenz, die ein Objekt auch über Adreßräume hinweg eindeutig identifiziert. Sie besteht aus einer Typidentifizierung und einer bestimmten Anzahl von Profilen (*Interoperability Profiles*, Abk. *IOPs*). Diese Profile enthalten eine Internet-Adresse, eine Port-Nummer und eine intern vom ORB erzeugte und verwendete Objektidentifikation (vom Standard als *object_key* bezeichnet). Der Inhalt dieses Objektschlüssels gilt als „verschleiert“ (*opaque*). Nur der ORB, der diese IOR erzeugt hat, kennt ihren genauen Inhalt.

Beispiel für eine IOR im Klartext:

```
TypeID:    "IDL:Hello:1.0"
Profiles:  1. IIOP 1.0 127.0.0.2 5001
"OB/ID+NUM.IDL:Hello:1.0.0"
```

Die TypeID ist die Typidentifizierung des Objekts. Sie kann durch die IDL-Direktive `#pragma` beeinflußt werden. Wie bereits beschrieben, enthalten die Profiles die Internet-Adresse (hier `127.0.0.2`), die Port-Nummer (hier `5001`) und die ORB-spezifische Objektidentifikation. Außerdem ist die Zeichenkette IIOP als Bezeichner und die Versionsnummer des IIOPs in den Profiles enthalten.

Eine IOR kann transient oder persistent sein. Letzteres bedeutet nicht, daß auch das dazugehörige CORBA-Objekt persistent ist. Sollte eine Server-Anwendung gestoppt und wieder gestartet worden sein, so verlieren alle transienten IORs ihre Gültigkeit. Bei persistenten IORs ist dies nicht der Fall. Eine persistente IOR hat den Vorteil, daß sie der Client-Anwendung nur ein einziges mal zugestellt werden muß. Dies setzt jedoch voraus, daß bei einem erneuten Start der Server-Anwendung die Internet Adresse des Rechners unverändert geblieben ist und die verwendete Portnummer frei ist, da beide Angaben in der IOR enthalten sind.

3. Ermittlung der Einstiegsreferenz

Ein Schwachpunkt des bestehenden CORBA-Standards ist die Art und Weise, wie ein Client an die IOR eines entfernten Objekts gelangt. CORBA definiert hierfür zwei Wege:

- die Verwendung der ORB-Operation `resolve_initial_references()` sowie
- die Benutzung der ORB-Operationen `object_to_string()` und `string_to_object()`.

Mit der Operation `resolve_initial_references()` hat CORBA einen *herstellerabhängigen* Mechanismus definiert, um an eine Einstiegsobjektreferenz zu gelangen. Dieser Weg ist normalerweise mit der Übergabe von Kommandozeilen-Parametern, der Erstellung von Konfigurationsdateien oder aber mit dem Setzen von Umgebungsvariablen verbunden und scheidet damit aus, wenn es sich um portable Anwendungen handeln soll. Derzeit können folgende Zeichenketten als Argumente dieser Operation verwendet werden:

- `RootPOA`,
- `POACurrent`,
- `InterfaceRepository`,
- `NameService`,
- `TradingService`,
- `SecurityCurrent`,
- `TransactionCurrent`,
- `DynAnyFactory`,
- `ORBPolicyManager`,
- `PolicyCurrent`,
- `NotificationService` und
- `TypedNotificationService`.

Die Operationen `object_to_string()` und `string_to_object()` stellen einen zwar portablen, aber wenig eleganten Weg dar, um *herstellerunabhängige* Anwendungen zu programmieren. Die erste Operation kann dazu verwendet werden, eine CORBA-

Objektreferenz in ihre Zeichenkettendarstellung zu konvertieren. Nach der Umwandlung kann sie beispielsweise in einer Datei oder Datenbank gespeichert werden. Die zweite Operation bildet das Gegenstück zu der zuvor genannten. Sie wird dazu verwendet, eine Referenz auf ein CORBA-Objekt aus ihrer Zeichenkettendarstellung in eine „richtige“ Referenz zu konvertieren. Die Frage, wie der Client dann an eine solche Zeichenkette gelangt, ist im Standard jedoch bisher nicht geklärt. Mögliche Übertragungswege sind z.B.:

- Auslesen aus einer Datei, auf die sowohl Client als auch Server zugreifen können (z.B. per NFS oder AFS),
- Übertragung per Dateitransfer (FTP),
- Übergabe als Kommandozeilen-Parameter oder
- Ermittlung über Umgebungsvariablen.

Abgesehen davon, daß diese Vorgehensweise der eigentlichen CORBA-Idee der Unabhängigkeit von Client und Server zuwiderläuft, stellt sie auch eine mögliche Fehlerquelle dar. Da zwischen der Erzeugung und dem Einlesen der IOR einige Zeit vergehen kann, ist es möglich, daß die IOR inzwischen nicht mehr gültig ist. Die Verfahren, die beispielsweise die Firmen Inprise (Agenten) oder Iona (Dämonen) anbieten, um an eine Einstiegsreferenz zu kommen, sind zwar komfortabler, aber nicht standardkonform. Die OMG sollte auch die Schnittstelle zu einem solchen Agenten bzw. Dämon standardisieren, um so das Problem der Ermittlung der Einstiegsreferenz zu lösen. Daneben können die zur Laufzeit verfügbaren Dienste mittels der ORB-Operation `list_initial_services` ermittelt werden. Das Ergebnis dieser Operation ist ein Feld aus Strings mit Namen der zu diesem Zeitpunkt bekannten Dienste.

In der aktuellen CORBA-Version 2.4 [2] wurde erstmals ein „*well-known*“ Port (Portnummer 683) definiert, was zu einer geringfügigen Erleichterung der Problematik der Ermittlung einer Einstiegsreferenz führt.

Die OMG-Spezifikation definiert außerdem zwei Basisdienste, mit deren Hilfe eine Anwendung an weitere Objektreferenzen gelangen kann: den *Naming Service* und den *Trading Service*.

4. Naming Service

Mit seiner Fähigkeit, verteilte Objekte zu lokalisieren, ist der *Naming Service* [4] der wohl fundamentalste CORBA-Dienst. Er wurde als einer der ersten Dienste von der OMG spezifiziert und inzwischen von nahezu allen ORB-Herstellern implementiert. Die Aufgabe des *Naming Services* ist mit der eines Telefonbuchs vergleichbar. Mit ihm lassen sich Objekte (genauer: Objektreferenzen) mit einem frei wählbaren Namen versehen. Die Zuordnung eines Namens zu einem Kontext bzw. einer Objektreferenz wird Bindung genannt. Ein Kontext entspricht hierbei einer Menge von Bindungen. Ein Name ist in seinem Kontext eindeutig und führt zu genau einem Kontext bzw. einer Objektreferenz.

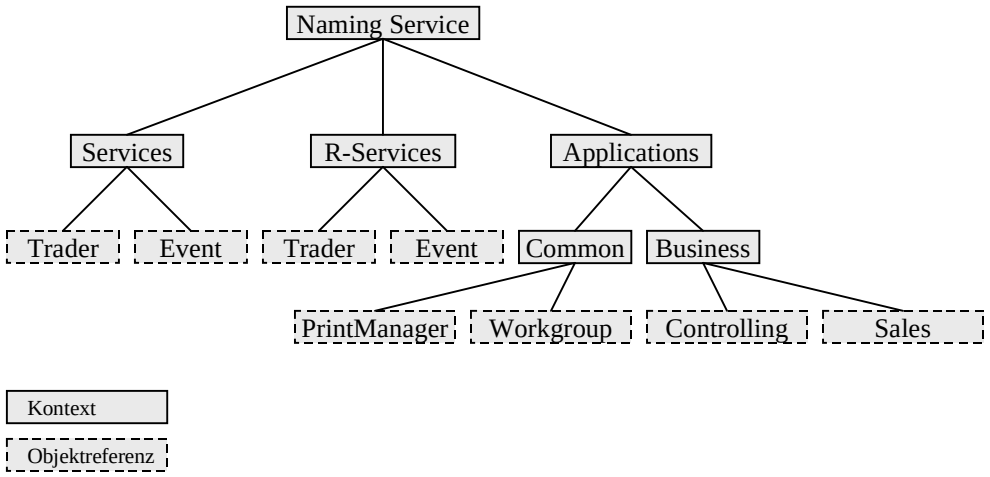

Abbildung 1: Ein Namensgraph

Der Namensdienst ist aus zwei Basisschnittstellen, dem `NameContext` und dem `BindingIterator`, zusammengesetzt. Der `NameContext` beinhaltet "benannte" Objektreferenzen. Er stellt eine Reihe von Operationen zur Verfügung, um CORBA-Objekte anzumelden (`bind`, `rebind`) bzw. abzumelden (`unbind`) oder die Referenz eines bestimmten CORBA-Objekts zu ermitteln (`resolve`). Außerdem bietet die `NameContext`-Schnittstelle die Möglichkeit, weitere `NameContext`-Objekte innerhalb des *Root*-`NameContexts` anzulegen (`bind_context`, `rebind_context`, `bind_new_context`, `new_context`), abzumelden (`unbind`), zu ermitteln (`resolve`) und zu löschen (`destroy`). Der `BindingIterator` wird benutzt, um zwischen verschiedenen Einträgen eines bestimmten Namenskontextes zu navigieren. Zu diesem Zweck stellt er die Operationen `next_one` und

`next_n` zur Verfügung. Um an die Referenz des Namensdienstes zu gelangen, kann eine Anwendung die ORB-Operation `resolve_initial_references ("NameService")` aufrufen. Das Ergebnis dieses Aufrufs ist vom Typ `CosNaming::NamingContext` und repräsentiert den initialen Kontext des Namensdienstes. Der folgende Code-Ausschnitt (Java) soll die Ermittlung einer Objektreferenz verdeutlichen:

```
try {
  ORB orb = ORB.init();
  org.omg.CORBA.Object obj =
    orb.resolve_initial_references ("NameService");
  NamingContext nc = NamingContextHelper.narrow(obj);
  NameComponent[] aName = new NameComponent[1];
  aName[0] = new NameComponent();
  aName[0].id = "PrintManager";
  aName[0].kind = "Color";
  org.omg.CORBA.Object aObj = nc.resolve(aName);
  //...
 }
 catch(SystemException ex) {
  System.exit(1);
}
```

Mit der Spezifikation des Interoperable Naming Service (INS) [3] hat die OMG auf die häufig geäußerte Kritik (u.a. [1]) an den bestehenden Möglichkeiten zur Ermittlung der Einstiegsreferenz reagiert. Der Aufbau des INS entspricht weitgehend dem des Namensdienstes, enthält jedoch einige Erweiterungen. Diese betreffen hauptsächlich:

- die Einführung neuer Adressierungsarten für CORBA-Objektreferenzen sowie
- die Standardisierung der Kommandozeilenoptionen.

Daneben wurde eine neue Schnittstelle (`NamingContextExt`) eingeführt, welche verschiedene Operationen zur Konvertierung der neu eingeführten Adressierungsarten in Objektreferenzen enthält.

4.1. CORBA URLs

Die neuen Adressierungsarten - sogenannte „CORBA URLs“ - sind an die WWW-Adressierung angelehnt. Die `corbaloc`-Adressierungsart unterstützt verschiedene Protokolle, wobei der Aufbau der URL protokollabhängig ist. Derzeit wurden zwei Standardprotokolle definiert: `iiop` und `rir`.

Die `corbaloc`-Adressierungsart für das IIOP-Protokoll besteht aus:

- dem Bezeichner `corbaloc`,
- dem Protokoll (als Standardprotokoll wird `iiop` verwendet und muß nicht explizit angegeben werden),
- der Protokollversion (Standardwert ist `1.0`),
- dem Rechnernamen oder der Rechneradresse,
- der Portnummer (als Standardport wird die Portnummer `2809`[1] verwendet) und
- einer Objektidentifikation.

Das allgemeine Format der `corbaloc`-Adressierungsart lautet:

```
corbaloc:[iiop]:[version@]host[:port]/object-key
```

Die Angabe mehrerer Lokalisierungsangaben ist möglich. Diese müssen durch Kommas voneinander getrennt werden. In diesem Fall verwendet der ORB die einzelnen Angaben der Reihe nach und versucht, die dazugehörige Objektreferenz zu ermitteln.

Angenommen, auf den Rechnern `chili.wifo.uni-mannheim.de` (Portnummer `2809`) und `thymian.wifo.uni-mannheim.de` (Portnummer `10000`) laufen zwei Namensdienste. Um an die Referenz eines der beiden Namensdienste zu gelangen, kann als URL:

```
corbaloc::chili.wifo.uni-mannheim.de,
:thymian.wifo.uni-mannheim.de:10000/NameService
```

verwendet werden. In diesem Fall wird zuerst versucht, die Objektreferenz zum Namensdienst auf dem Rechner `chili` (Standardport) zu ermitteln. Mißlingt es, so versucht der ORB die

Referenz zu dem auf dem Rechner `thymian` laufenden Namensdienst (Portnummer `10000`) zu ermitteln.

Das `rir`-Protokoll dient als Abkürzung für die ORB-Operation `resolve_initial_references`.. Der allgemeine Aufbau der `corbaloc`-Adressierungsart für das `rir`-Protokoll lautet:

```
corbaloc:rir:[/id]
```

wobei `id` für den Bezeichner des Dienstes steht. Ohne die `id`-Angabe wird standardmäßig der Bezeichner `NameService` benutzt.

Die `corbaname`-Adressierungsart beruht auf einer anderen Semantik. Im Unterschied zum `corbaloc`-Schema beinhaltet das `corbaname`-Schema die Zuordnung der Objektreferenz innerhalb eines bestimmten `NameContexts`. Die Lokalisierungsangaben beziehen sich in diesem Fall nicht auf die gesuchte Objektreferenz, sondern auf die Objektreferenz zum Namensdienst. Die Objektidentifikation beschreibt nicht das gesuchte Objekt selbst, sondern enthält den Pfad innerhalb des Namensgraphen, der zur gesuchten Objektreferenz führt.

Tabelle 2 enthält eine Zusammenfassung der verschiedenen Adressierungsarten.

Schema	**Darstellung**	**Art**
`IOR:`	standardisiertes Zeichenkettenformat der IOR	erforderlich
`corbaloc:`	IIOP-spezifisches Zeichenkettenformat der IOR	erforderlich
`corbaname:`	IIOP Namensdienst-spezifische URL	erforderlich
`file:`	spezifiziert eine Datei mit einer URL/IOR	optional
`ftp:`	die angegebene Datei spezifiziert eine URL/IOR, die über das FTP-Protokoll aufrufbar ist	optional
`http:`	die angegebene Datei spezifiziert eine URL/IOR, die über das HTTP-Protokoll aufrufbar ist	optional

Tabelle 2: Mögliche Adressierungsarten von Objektreferenzen

[1] Vorschlag von IANA

4.2. Die standardisierten Kommandozeilen-Optionen

Die zweite Verbesserung betrifft die Standardisierung der Kommandozeilen-Optionen. Mit Hilfe des Arguments `-ORBInitRef` kann beim Start der CORBA-Anwendung dieser eine Einstiegsreferenz mitgeteilt werden. Die allgemeine Syntax lautet hierfür:

```
-ORBInitRef <ObjectID>=<ObjectURL>.
```

Die `ObjectID` entspricht hierbei dem Namen, den die Anwendung beim Aufruf der ORB-Operation `resolve_initial_references` als Argument übergeben würde.

Mögliche Anwendungsbeispiele dafür lauten:

```
-ORBInitRef NameService=IOR:00230021AB...
-ORBInitRef NameService=
    corbaloc::chili.wifo.uni-mannheim.de/NameService
-ORBInitRef TradingService=
    corbaname::1.1@chili.wifo.uni-mannheim.de:10000/Services/Trader
```

Daneben wurde die Kommandozeilenoption `-ORBDefaultInitRef` spezifiziert. Im Unterschied zu der zuvor beschriebenen braucht hier der Name (`ObjectID`) nicht angegeben werden. Somit kann sie zu Ermittlung von Objektreferenzen verwendet werden, deren Namen beim Start der Anwendung (noch) nicht bekannt sind.

5. Trading Service

In offenen, heterogenen, verteilten Systemen ist das namensbasierte Auffinden eines Dienstanbieters keine ausreichende Lösung, da der Dienstnutzer den exakten Namen seines Kommunikationspartners kennen muß. Eine bessere Lösung kann eine auf Diensteigenschaften basierte Suche und anschließende Auswahl liefern.

Ziel des *Tradings* ist die Vermittlung eines Dienstangebots an einen Dienstnutzer. Die Nutzer des Trading Services [5] lassen sich in zwei Kategorien einteilen: Dienstanbieter (*Exporters*) und Dienstnutzer (*Importers*). Diese Einteilung ist nicht fest, d.h. ein Dienstanbieter kann genauso gut als Kunde auftreten.

Ein Dienstanbieter kann unter Angabe seiner Zugangsinformationen, eines Diensttyps und verschiedener Attribute sein Angebot exportieren. Ein Attribut repräsentiert ein Merkmal eines Dienstes. Es besteht aus einen Attributnamen und einem Attributtyp. Attribute können statisch und dynamisch sein. Statische Attribute dienen zur Darstellung der Eigenschaften eines Dienstes, während dynamische Attribute den aktuellen Zustand eines Dienstes wiedergeben. Nach der Anmeldung beim *Trader* kann der Exporter auf die Anfragen potentieller Dienstnutzer reagieren.

Ein Dienstnutzer kann unter Angabe eines gewünschten Diensttyps und der für ihn relevanten Attribute die erforderlichen Zugangsinformationen ermitteln. Der vermittelnde Trader kann kein, ein oder mehrere Angebote zurückliefern. Dabei überprüft er alle bei ihm angemeldeten Dienstanbieter auf Konformität zum verlangten Diensttyp und liefert dem Importer alle in Frage kommenden Bindungen. Der Trader setzt bei seiner Suche verschiedene Strategien (*Polices*) ein. Diese können vom Importer verändert werden, so daß dieser die Gesamtanzahl der zurückgegebenen Angebote oder deren Reihenfolge beeinflussen kann.

Die Anzahl der gefundenen Angebote kann unter Umständen sehr groß ausfallen. Der Trading Service erlaubt es die Anzahl der Ergebnisse der Suche durch die Angabe verschiedener Strategien (*Policies*), Einschränkungen (*Constrains*) und Einstellungen (*Preferences*) zu begrenzen.

Den Kern der Trader-Spezifikation bilden die folgenden Schnittstellen:

- `Lookup`: Diese Schnittstelle wird von Dienstnutzern und anderen Tradern benutzt um Dienste aufzufinden. Sie enthält nur eine einzige Operation (`query`).
- `OfferIterator`: Mit Hilfe dieser Schnittstelle kann ein Dienstnutzer durch verschiedene Dienstangebote navigieren.
- `Register`: Dienstanbieter können diese Schnittstelle benutzen, um ihre Dienste zu offerieren oder zurückzunehmen.
- `Link`: Diese Schnittstelle ermöglicht es einem Trader, die Dienste anderer Trader in Anspruch zu nehmen. Auf diese Weise kann ein verteilter Trader-Verbund aufgebaut werden.
- `Proxy`: Um an die Objektreferenz eines bestimmten Dienstanbieters zu gelangen, muß ein Trader diese Schnittstelle benutzen.
- `DynamicPropEval`: Mit Hilfe dieser Schnittstelle kann der Trader die aktuellen Werte der dynamischen Attribute eines Dienstanbieters jederzeit ermitteln.

Daneben steht eine Reihe weiterer Schnittstellen z.B. für administrative Aufgaben, zur Verfügung.

Die Referenz eines Traders kann mittels der ORB-Operation `resolve_initial_references("TradingService")` ermittelt werden. Das Ergebnis dieses Aufrufs ist vom Typ `CosTrading::Lookup`.

6. Zusammenfassung

Durch die Einführung einer eindeutigen Objektreferenzierung auf Basis der so genannten *Interoperable Object References* (IORs) und eines standardisierten Übertragungsprotokolls, dem *Internet-Inter-ORB-Protocol* (IIOP), im Rahmen von CORBA 2.0 können Anwendungen, die auf verschiedenen ORB-Produkten basieren, miteinander kommunizieren. Daneben spezifiziert die OMG eine Reihe von Diensten, welche die Grundfunktionalität eines ORBs um generelle, systemnahe Erweiterungen, die für die grundsätzliche Funktionsweise einer verteilten Anwendung von Bedeutung sind, ergänzen. Um an die Referenzen weiterer verteilter Objekte zu gelangen, können der *Naming Service* oder der *Trading Service* verwendet werden. Während der Naming Service für das namensbasierte Auffinden von Objekten zum Einsatz kommen kann, ermöglicht der Trading Service eine auf Diensteigenschaften basierte Suche und anschließende Auswahl.

Bis zur CORBA-Version 2.3 stellte die Art und Weise, wie man an die Einstiegsreferenz (z.B. des Naming Services) gelangt, einen großen Schwachpunkt und eine mögliche Fehlerquelle dar. So mußte die Referenz entweder aus einer Datei ausgelesen werden (z.B. falls ein Zugriff über NFS oder AFS möglich war) oder gar erst mit FTP übertragen werden. Da die Zeit zwischen der Erzeugung und dem Einlesen der IOR (aus einem Dateisystem) unterschiedlich lang sein kann, ist es nach dem Einlesen möglich, daß die IOR inzwischen nicht mehr gültig ist. Mit der Spezifikation des *Interoperable Naming Services* (INS) sowie der Einführung neuer Adressierungsarten für CORBA-Objektreferenzen und der Standardisierung der Kommandozeilenoptionen hat die OMG auf die häufig geäußerte Kritik an den unzureichend standardisierten Möglichkeiten zur Ermittlung der Einstiegsreferenz reagiert. Vor allem durch die letzten beiden Punkte wurde die Portabilität CORBA-basierter Anwendungen deutlich verbessert.

Herstellerspezifische Verfahren, die beispielsweise Visigenic (Agenten) oder Iona (Dämonen) anbieten, um an eine Einstiegsreferenz zu kommen, sind zwar aus der Sicht des Programmierers komfortabel, aber nicht standardkonform. Die OMG sollte sich überlegen, ob nicht auch die Standardisierung der Schnittstelle zu einem solchen Agenten bzw. Dämon sinnvoll wäre, um dem Programmierer die Entwicklung verteilter Anwendungen noch weiter zu erleichtern.

Abkürzungsverzeichnis

CDR	Common Data Representation
CORBA	Common Object Request Broker Architecture
ESIOP	Environment-Specific Inter-ORB Protocol
GIOP	General Inter-ORB Protocol
IDL	Interface Definition Language
IIOP	Internet Inter-ORB Protocol
IOP	Interoperability Profile
IOR	Inter-ORB Reference
IP	Internet Protocol
OMA	Object Management Architecture
OMG	Object Management Group
ORB	Object Request Broker
OSF	Open Software Foundation
TCP	Transmission Control Protocol

7. Literatur

[1] Aleksy, M., Schader, M., Tapper C. (1999): “Interoperability and Interchangeability of Middleware Components in a Three-Tier CORBA-Environment – State of the Art”, in: Proceedings Third International Enterprise Distributed Computing Conference EDOC’99, 27-30 Sept. 1999, University of Mannheim, Germany, IEEE, Piscataway, New Jersey, S. 204-213

[2] Object Management Group (2000): “CORBA/IIOP 2.4 Specification”; OMG Technical Document Number 00-10-01, ftp://ftp.omg.org/pub /docs/formal/00-10-01.pdf

[3] Object Management Group (2000): “Interoperable Naming Service Specification“; OMG Technical Document Number 00-11-01, ftp://ftp.omg.org/pub/docs/formal/00-11-01.pdf

[4] Object Management Group (2000): “Naming Service Specification“; OMG Technical Document Number 00-06-19, ftp://ftp.omg.org/pub/docs/formal/00-06-19.pdf

[5] Object Management Group (2000): “Trading Service Specification“; OMG Technical Document Number 00-06-27, ftp://ftp.omg.org/pub/docs/formal/00-06-27.pdf

[6] Schader M., Aleksy M., Tapper C. (1998): „Interoperabilität verschiedener Object Request Broker nach CORBA2.0-Standard“; in: OBJEKTspektrum, 3/98, S. 72-77, http://www.wifo.uni-mannheim.de/ IIOP