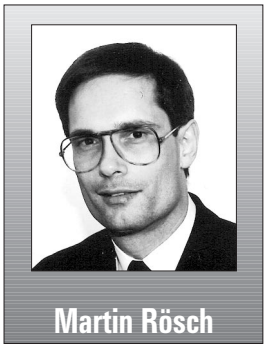
OMG-Standards und ihre Bedeutung für die Praxis



Martin Rösch

Der Artikel beschreibt CORBA (Common Object Request Broker Architecture), den ersten Standard der OMG, seine Bedeutung für die Praxis, die vorhandenen Produkte und das Verhältnis von COBRA zu Microsoft OLE und OSF/DCE.

Object Management Group (OMG)

1989 gegründet, hat die OMG in nur vier Jahren praktisch die gesamte Informationsindustrie hinter sich versammelt. "Integration von Anwendungen" ist das Ziel der OMG, deren 350 Mitglieder sich einig sind, daß dieses Ziel heute am besten mit Hilfe von Objektorientierung erreicht werden kann.

Unzufriedenheit über die monolithische Struktur und schlechte Kombinierbarkeit heutiger Softwaresysteme haben bei der Gründung der OMG ebenso Pate gestanden wie die begründete Annahme, daß die Computersysteme der Zukunft mit den heutigen Programmiermethoden nicht mehr entwickelt werden können.

Die OMG will durch ihre Standards eine Softwarefertigung ermöglichen, die mit der Effizenz und Qualität industrieller Prozesse vergleichbar ist. Dabei ist die Verteilbarkeit von Programmen und Daten eher ein Abfallprodukt als ein zentrales Thema der OMG-Arbeit.

Object Management Architecture (OMA)

Als Rahmenwerk, dessen Komponenten durch definierte Schnittstellen verbunden werden, beschreibt die OMG ihre Architektur. Sie ist im "Object Management Architecture Guide" [OMG90] beschrieben und besteht aus vier verschiedenen Komponenten bzw. Arten von Komponenten (siehe Abb. 1), nämlich:

-Object Request Broker,
-Object Services,
-Common Facilities,
-Application Objects.

Der Object Request Broker (ORB) ist der Postverteiler, der Verarbeitungsanforderungen (Requests) an Objekte weiterleitet. Er ist bereits standardisiert (CORBA)

Object Services sind Dienste, die von jedem Objekt bereitgestellt werden müssen wie z.B. Erzeugen neuer Objekte, Löschen, Benachrichtigungen über eingetretene Ereignisse usw. Die Standardisierung der Object Services steht kurz vor der Fertigstellung.

Common Facilities sind Verarbeitungen, die für viele Objekte nützlich sind, aber nicht in jedem Fall gebraucht werden. Beispiele hierfür sind Drucken, Wiedervorlagen, E-Mail-Anschluß, Bildschirmanzeige usw. Die Standardisierung der Common Facilities hat gerade begonnen.

Application Objects sind die eigentlichen Arbeitstiere der Objektwelt. Sie erledigen die fachliche Arbeit. Ein Beispiel ist die eben erstellte Rechnung, die jetzt gedruckt werden soll. Weitere Objekte sind die bereits vorhandenen Rechnungen. Und die Kunden, an die sie gestellt wurden. Die OMG wird Application Objects nicht standardisieren. Hier sind Branchenvereinigungen und auch die EDI-Standardsierungsgremien gefordert.

CORBA-Standard

Im Januar 1992 wurde CORBA 1.1 als erster OMG-Standard veröffentlicht [OMG 92]. Die OMG hatte diese wichtige Einigung nur zweieinhalb Jahre nach ihrer Gründung herbeigeführt.

Motivation

Mit dem CORBA-Standard wird das Ziel verfolgt, die Abhängigkeit zwischen verschiedenen Teilen eines Systems auf das absolute Minimum zu reduzieren. Das bedeutet, daß technische Abhängigkeiten zwischen Anforderer (Client) und Bereitsteller (Server) eines Service vollständig aus den Anwen-

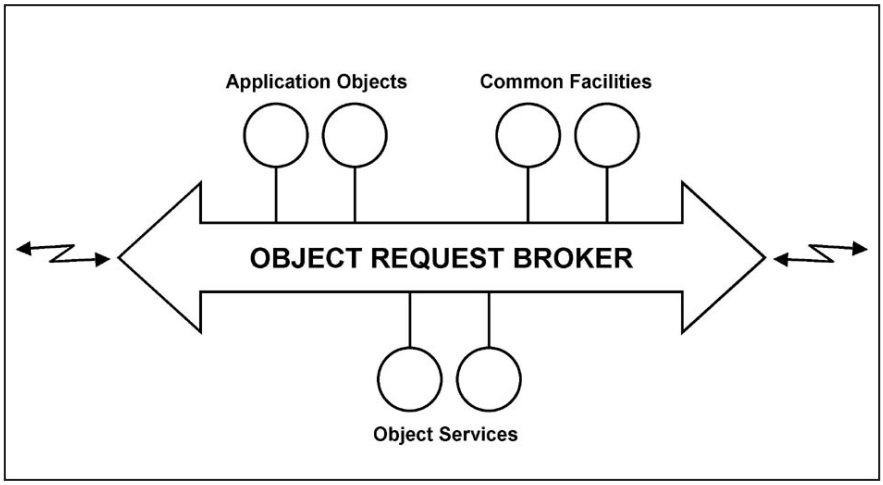


Abb. 1: Komponenten und Arten von Komponenten der Object Management Architecture

Martin Rösch ist Diplom-Informatiker und leitet die Object Academy. Object Academy ist eine Dienstleitung von Rösch Consulting Gesellschaft für innovative Softwareentwicklung GmbH, Kaarst bei Düsseldorf.

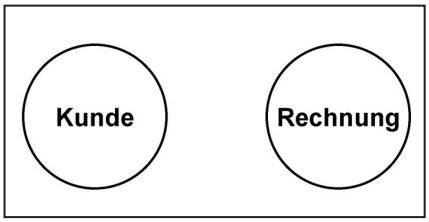


Abb. 2: Objekttypen

dungsprogrammen verschwinden und nur noch solche Abhängigkeiten übrigbleiben, die fachlich bedingt und deshalb unverzichtbar sind. So weiß z.B. eine Rechnung, wer ihr zugehöriger Kunde ist. Doch wenn sie die Adresse des Kunden benötigt, holt sie die Adresse mit dem gleichen Programmbefehl – unabhängig davon, wo die Daten des Kunden liegen und auf welcher Maschine die Verarbeitung für die Bereitstellung der Adresse durchgeführt wird. Die Rechnung muß hierfür nur die Nachricht "Adresse holen" an ihren Kunden schicken. Das Zustellen der Nachricht, das Auslösen der Verarbeitung und den Rücktransport des Ergebnisses übernimmt der ORB.

Diese Unabhängigkeit bedeutet, daß Sender und Empfänger einer Nachricht nicht mehr auf der gleichen Maschine oder im gleichen Adreßraum liegen müssen. Die Konsequenz: Das Überschreiten dieser Grenzen (Maschine, Adreßraum, ...) ist im Programmcode nicht mehr sichtbar. Dadurch wird Client/Server-Programmierung erheblich vereinfacht.

Die Unabhängigkeit bedeutet auch, daß die verschiedenen Teile eines Systems aus verschiedenen Entwicklungsabteilungen stammen können. Die Teile können zugekauft oder vorgefertigt sein. Damit wird, was für die industrielle Fertigung seit Jahrzehnten selbstverständlich ist, jetzt auch für die Software möglich.

Einbettung in die Object Management Architecture (OMA)

Object Request Broker sind eine Komponente der Object Management Architecture der OMG. Sie sind der Kern der Architektur, da die anderen Komponenten bzw. Arten von Komponenten zum Teil den ORB für ihre Kommunikation untereinander benötigen.

CORBA-Objekte und Typen

CORBA vermeidet bewußt den Begriff "Klasse", da er in fast jeder Programmiersprache eine eigene Bedeutung hat und daher zu Mißverständnissen geradezu einlädt. Stattdessen werden Objekte bei CORBA zu Typen zusammengefaßt (siehe Abb. 2). Ein Beispiel: Wir können festlegen, daß alle unsere Rechnungen zum Typ "Rechnung" gehören. Alle Objekte eines Typs haben ein gemeinsames Interface (siehe Abb. 4).

Darüber hinaus gibt es im CORBA-Standard auch "Basistypen" wie z.B. Integer, Strings, Float usw. und "Zusammengesetzte Typen" wie z.B. Gruppen (structs) und Redefinitionen (unions), die für die Beschreibung der Parameter von Services benötigt werden.

Requests

Wenn ein Client einen Service eines Objektes nutzen möchte, schickt er eine Nachricht an das Objekt. Das Schicken der Nachricht wird bei CORBA als "Request" bezeichnet.

Ein Objekt, das gerade einen Request erfüllt, kann selbst auch wieder gegenüber anderen Objekten als Client auftreten. Beispiel: Die Rechnung, die in Abb. 3 den Service "Drucke Dich" ausführt, wird zum Client ihres Kunden, weil sie von ihm die Adresse benötigt. Auch herkömmliche Programme können als Clients von Objekten auftreten.

Interfaces

Jedes Objekt hat eine Schnittstelle (Interface). Unser Rechnungs-Objekt hat z. B. das Interface "Rechnung". Das Interface beschreibt alle Services., die von einem Objekt zur Verfügung gestellt werden. Ein Interface entspricht der Klassenschnittstelle, wie sie auch von OOPs bereitgestellt wird. Nur die im Interface beschriebenen Services können von Clients durch Requests abgerufen werden. Services, die nicht im Interface stehen, sind von außen nicht erreichbar. Sie können nur "privat" vom Eigentümer-Objekt benutzt werden.

Interfaces können in einer Vererbungsbeziehung stehen. Beispiel: Das Interface "Rechnung" erbt vom Interface "Geschäftspapier". Mehrfachvererbung ist möglich.

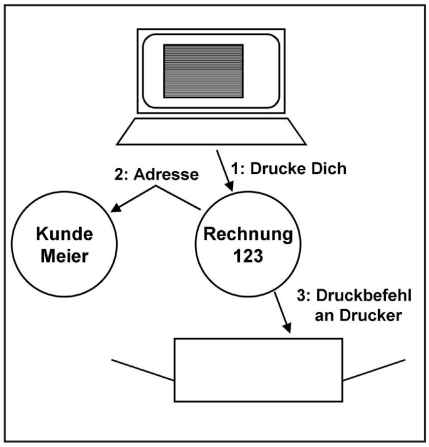


Abb. 3: Objekte vom Type „Rechnung" sind Clients ihrer zugehörigen Objekte vom Typ „Kunde". Diese Objektstruktur ist gleichzeitig auch die Architektur des fertigen Systems.

IDL = Interface Definition Language

Die Beschreibung von Interfaces ist durch die IDL von CORBA standardisiert. IDL ist stark an C++ angelehnt.
Beispiel: Rechnungen können sich drucken, anzeigen, Zahlungseingänge registrieren und mahnen (siehe Abb. 4). IDL beschreibt die Namen und die Parameter der Services. IDL unternimmt keinen Versuch, die durchgeführte Verarbeitung genauer zu beschreiben. Aus der IDL eines Interface generiert der IDL-Compiler für jeden aufrufbaren Service des Interface ein kleines Programm, den Client-Stub, der anschließend von Clients benutzt werden kann, um den zugehörigen Request abzuschicken. Damit hat jeder Client den Eindruck, der angeforderte Service werde durch ein lokales Unterprogramm erledigt.

Implementierungen

Im Gegensatz zum Interface eines Typs, das für alle Objekte des Typs identisch ist – unabhängig von Maschinen, Betriebssystemen und Sprachen – ist die Implementierung des Objektverhaltens an die Maschine gebunden, auf der der Service durchgeführt wird.
Beispiel: Wenn unsere Rechnungen auf einem MVS/CICS-System und auf einer HP-Workstation leben und arbeiten sollen, müssen wir zwei Implementierungen für Objekte vom Typ "Rechnung" haben, nämlich eine für MVS/CICS und die andere für HP-UX.

Verhältnis von CORBA zu Microsoft OLE

Microsoft ist Mitglied der OMG, beteiligt sich aber nicht am Standardisierungsverfahren. Microsoft erweckte sogar bis vor kurzem den Eindruck, als sollte das hauseigene OLE (Object Linking and Embedding) gegen CORBA durchgesetzt werden.

Nach der Einschätzung des Autors handelt es sich dabei um einen Teil einer Marktbeherrschungsstrategie: Microsoft wird, sobald die eigene verteilte Technik funktioniert, auf den CORBA-Zug aufspringen. Doch wenn Microsoft dies heute schon zugäbe, würden viele Entwickler auf SOMobjects von IBM wechseln, das (lt. IBM-Ankündigung) den CORBA-Standard unter Windows implementieren wird.

Es gibt Anzeichen dafür, daß Microsoft die bisherige abwartende Haltung gegenüber der OMG und dem CORBA-Standard aufgibt und zu einer kooperativeren Einstellung findet. Wie DEC, der Allianzpartner von Microsoft, ankündigte wird OLE eine CORBA-Oberfläche erhalten. Das bedeutet, daß

```
Interface Rechnung{
        void drucken();
        void anzeigen();
        void Zahlungseingang_registrieren(in DM Betrag);
        void mahnen();
}
```

Abb. 4: Die Interface-Beschreibung für Objekte vom Typ Rechnung

OLE den grundlegenden Transportmechanismus im Inneren der CORBA-Lösung von DEC/Microsoft bilden wird, während nach außen nur die CORBA-Schnittstelle zu sehen ist. Anwendungsprogramme können dann OLE benutzen, ohne davon abhängig zu sein.

Vielleicht hat aber auch Lotus Development mit der angekündigten CORBA-Unterstützung seiner Produkte den Denkprozeß bei Microsoft beschleunigt.

Verhältnis von CORBA zu OSF/DCE

Anders als OLE ist Distributed Computing Environment von der Open Software Foundation (OSF/DCE) ein offenes System. OSF/DCE ist einige Jahre älter als CORBA und verfolgte ein bescheideneres Ziel. Mit DCE sollte die Technik des Fernaufrufs (Remote Procedure Call, kurz RPC) herstellerunabhängig und netzweit zur Verfügung gestellt werden. Mit RPC können Unterprogramme auf anderen Maschinen genauso einfach abgerufen werden wie auf der lokalen Maschine. RPC macht den Unterschied zwischen lokal und nicht lokal vorhandenen Unterprogrammen für Anwendungsentwickler unsichtbar.

DCE kennt keine Objekte, bietet aber eine ideale Grundlage für die Implementierung von Object Request Brokern nach dem CORBA-Standard. Mindestens zwei Hersteller von ORBs, IBM und Hewlett Packard, haben bereits angekündigt, daß sie ihre ORB-Produkte auf der Basis von DCE implementieren werden.

Sollte man deshalb DCE auslassen und auf CORBA warten? Nein, Sie verbauen sich nichts, wenn sie mit DCE beginnen. Achten Sie aber darauf, daß die Architektur Ihrer neuen Systeme die Möglichkeiten von CORBA berücksichtigt und sich auch die technische Realisierung im Detail an CORBA orientiert. Dann ist der Vorteil von DCE größer als der Umstellungsaufwand.

Vorhandene Produkte

Verschiedene Firmen bieten bereits CORBA-Produkte an (siehe Abb. 5):

DOMS von Hyperdesk und ACA von DEC waren die ersten Produkte, die sich am CORBA-Standard orientierten. DOMS von Hyperdesk läuft auf Unix und kann PC-Clients unterstützen. ACA von DEC lauft auf DEC-Rechnern und kann eine sehr breite Palette von Client-Plattformen (z.B. auch Windows und Macintosh) mit ORB-Funktionalität versorgen

Distributed Smalltalk von HP war der erste ORB, der die Verteilung von Objekten über mehrere Maschinen ermöglichte.

SOMobjects von IBM kann auf OS/2 und AIX eingesetzt werden. Die Erweiterung auf AS/400, MVS und MS-Windows 3.1 ist vorgesehen. Im Zusammenhang mit DSOM sind ebenfalls (mit Einschränkungen) verteilte Objekte möglich.

Orbix von IONA ist ein jüngerer ORB aus Irland, der auf UNIX läuft.

ASM von IBM war zunächst gar nicht als ORB vorgesehen, erfüllt aber die ORB-Funktionalität z.T. besser als das "richtige" ORB-Produkt der IBM (d.h. SOM). ASM und SOMobjects werden voraussichtlich 1994 in einem Produkt vereinigt werden.

Darüber hinaus gibt es eine Reihe von angekündigten Produkten:

ORBPlus von HP wird der erste ORB sein, der (mit Hilfe von Encina bzw. CICS) eine Transaktionssicherung von Objektverarbeitungen über Maschinengrenzen hinweg ermöglicht.

AppWare von Novell verbindet ORB-Technik mit visueller Programmierung durch die Einbindung der Serius-Produktlinie.

RC-ORB ist nicht für den Produktionseinsatz gedacht, sondern hilft bei der Ausbildung von Entwicklern.

Distributed Objects Everywhere (DOE) von SunSoft ist aus der gemeinsam mit HP entwickelten CORBA-Implementierung Distributed Object Management Facility (DOMF) entstanden.

Praktische Bedeutung von CORBA

Die Vorteile der Objektorientierung waren bisher nur innerhalb einzelner Programme (z.B C++) oder innerhalb einzelner Maschinen (z.B. bei Smalltalk) nutzbar. Objekte auf verschiedenen Maschinen konnten nicht direkt miteinander kommunizieren.

Darüber hinaus waren Objekte immer an eine bestimmte Programmiersprache gebunden. Beispiel: Smalltalk-Objekte können nicht mit C++ bearbeitet werden und umgekehrt. Sogar zwischen C++-Compilern bestehen Verständnisprobleme.

Mit beiden Einschränkungen räumt CORBA auf: Objekte sind nicht mehr auf eine Maschine beschränkt, sondern können ihre Dienste netzweit zur Verfügung stellen. Dabei können die Programmiersprachen, Betriebssysteme usw. von Client und Objekt-Implementierung unterschiedlich sein.

Maximale Unabhängigkeit

Als Ergebnis des CORBA-Einsatzes entsteht eine maximale Unabhängigkeit der einzelnen Objekte voneinander. Da nur noch die fachlichen Abhängigkeiten (z.B. muß eine Rechnung einen Kunden haben, der sie bezahlt) übrig bleiben, ist das absolute Minimum an

Verfügbare ORB-Produkte:

Hyperdesk	DOMS: Distributed Object Management System
DEC	ACA: Application Control Architecture
IBM	SOMobjects mit SOM und DSOM: Distributed System Object Model
IONA	Orbix
HP	Distributed Smalltalk
IBM	ASM: Application Services Manager*)

***) ASM erfüllt den Geist, aber nicht den Buchstaben des CORBA-Standards. Das soll sich 1994 durch Hinzufügen einer CORBA-Schnittstelle ändern.**

Angekündigte ORB-Produkte:

HP	**ORBPlus**
Novell	**AppWare**
Rösch Consulting	**RC-ORB**
SunSoft	**DOE**

Abb. 5: ORB-Produkte

Abhängigkeit erreicht und damit ein Maximum an Unabhängigkeit.

Bausteinarchitektur
Die mit CORBA erzielte maximale Unabhängigkeit der Komponenten betrieblicher Softwaresysteme ermöglicht, daß Änderungen eines Bausteins nur minimale Auswirkungen auf andere Bausteine haben [Rös93a]. Genauer gesagt, nur, wenn das Interface von Objekten geändert wird, können andere, benachbarte Objekte betroffen sein. Änderungen, die das Interface stabil lassen, bleiben nach außen unsichtbar.

Software-Komponenten-Markt
Die maximale Unabhängigkeit der Bausteine besteht sowohl für hausintern erstellte CORBA-Software als auch für zugekaufte CORBA-Komponenten. Damit öffnet CORBA die Türen für den Software-Komponenten-Markt. Erste Anfänge (mit ObjectWare von NeXT) führten schon 1992 trotz erheblicher Einschränkungen (kein CORBA, Bindung an NeXT-Hardware und an Objective-C) zu einem Angebot von über 100 Komponenten. Bei der IBM-Ankündigung der CORBA-Implementierung SOMobjects für OS/2 und AIX im Juni 1993 gab es bereits sieben Komponentenhersteller, die die Verfügbarkeit ihrer Software Komponenten für SOMobjects bekanntgaben, darunter auch Workflow-Manager, Kommunikations-Bausteine und Finanz-Software.

(Vorbereitung für) Dezentralisierung
CORBA bedeutet aber keinen Zwang zu verteilten Systemen. Die Nicht-Verteilung, bzw. die „Verteilung" aller Verarbeitungen auf einem einzigen Großrechner mit MVS/CICS oder IMS/TM ist ein Sonderfall eines verteilten Systems, der sogar noch Vorteile bei der Transaktionsabsicherung bietet.

CORBA bietet einen zusätzlichen Vorteil. Wenn später einmal doch verteilt werden soll, müssen CORBA-Anwendungsprogramme hierfür nicht mehr geändert werden. Ein erneutes Linken reicht aus, da nur die Client-Stubs ausgetauscht werden müssen.

Einsatz in der Praxis
CORBA-Produkte stehen bisher nur in begrenztem Umfang zur Verfügung. Doch sollte dies kein Grund für Untätigkeit sein, denn die meisten Vorarbeiten für den CORBA-Einsatz können schon lange vor der Installation eines ORB durchgeführt werden.

Was ist ein CORBA-Objekt?
Diese Frage klingt harmlos, ist aber von großer Bedeutung für den praktischen Erfolg der Implementierung. Die Antwort ist genauso einfach, doch fordert sie von Entwicklern oft erhebliches Umdenken beim Systementwurf, da nicht mehr technische Gegebenheiten, sondern die Begriffe der Fachabteilungen, ihre Konzepte und Denkweisen die oberste Schicht der Systemarchitektur, die CORBA-Objekte, bestimmen (siehe Abb. 3). Für die Festlegung der geeigneten Objekte ist eine objektorientierte Analyse (OOA) am besten geeignet.

Objektidentifikation
Die systemweite Identifikation von Objekten ist im ersten CORBA-Standard offen gelassen worden, so daß sich jeder Implementierer selbst darum kümmern muß. Doch zum Glück zeichnet sich auch hier eine industrieweite Einigung ab. Sowohl HP und IBM als auch andere ORB-Hersteller verwenden die UUIDs (Universally Unique Identifier) von OSF/DCE in ihren kommenden Produkten. Anwender, die bereits mit Vorbereitungen für eine eigene CORBA-Architektur beginnen möchten, sind gut beraten, das gleiche zu tun.

Client-Stubs sofort
Wer heute schon die Vorteile von CORBA in seinen Informationssystemen nutzen möchte, kann sofort damit beginnen. Das Warten auf ORB-Produkte ist nicht erforderlich, denn ein wesentlicher Teil von CORBA ist das Abschicken von Serviceanforderungen an Objekte mit Hilfe standardisierter Unterprogramme, sogenannter Client-Stubs, die in einem früheren Abschnitt beschrieben sind. Aus der Sicht der Programme, die die Services anfordern, sind Client-Stubs ganz normale Unterprogramme mit einigen Konventionen zur Parameterbenutzung. Wer diese Konventionen schon heute einhält, kann z.B. auch unter CICS, IMS oder BS2000 eine objektorientierte Architektur mit produktionsfähiger Leistung realisieren und wird außerdem kaum Probleme mit der Umstellung auf CORBA haben.

ORB-Produkte, sobald verfügbar und sinnvoll einsetzbar
Nicht in allen Umgebungen werden heute schon ORB-Produkte angeboten. Doch alle großen Hersteller arbeiten fieberhaft daran, ihren Anschluß an die CORBA Objektwelt möglichst bald anbieten zu können. Doch springen Sie nicht auf das erste Beta-Release, denn auch die Softwareentwickler großer Hersteller haben manchmal lange Lernkurven. Und wie gesagt, es gibt genug vorzubereiten.

Die nächsten Standards der OMG

Die nächsten Themen, die auf der Standardisierungsliste der OMG stehen, sind eine verbesserte und weitergeschriebene Version des CORBA Standards sowie die in zwei Gruppen unterteilten „Object Services".

CORBA 2.0
Das wichtigste Thema, das beim ersten CORBA-Standard (1.1 vom Januar 1992) offen blieb, war die Zusammenarbeit (Interoperabilität) verschiedener Object Request Broker. Die Zusammenarbeit von Object Request Brokern bedeutet für Anwender, daß verschiedene Objekt Systeme zusammengestöpselt werden können wie die vielzitierten Lego-Bausteine.

Andere Themen des CORBA 2.0-Standards [Rös93b] wie die Transaktionsabsicherung bei maschinenübergreifenden Objekt-Verarbeitungen und die einheitliche Identifikation von Objekten sind Folgen der Forderung nach Interoperabilität.

Ein anderes Gebiet der Forderungen von CORBA 2.0 betrifft Multimedia-Anwendungen. Hier müssen verschiedene Datenströme parallel und koordiniert geführt werden. Daraus leiten sich Sychronisationsservices und die zeitgleiche Benachrichtigung verschiedener Objekte ab.

Und schließlich ist es absehbar, daß die OMG ein Zertifizierungslabor einrichten wird, das mit Hilfe standardisierter Test-Suiten die Einhaltungen der OMG-Standards neutral überprüfbar macht.

Da die OMG ihre Standards nur auf der Basis vorhandener oder absehbarer Technologie formuliert, gab es im Dezember 1993 bereits drei Allianzen von Herstellern, die sich am „Aufgallop" zum Standard CORBA 2.0 beteiligten:

HP und IBM waren bereits im Juni 1993 übereingekommen, sich gegenseitig die bereits entwickelte Objekt-Technik (SOM von IBM und DOMF von HP) zur Verfügung zu stellen, um gemeinsam eine durchgängige verteilte Objektwelt über IBM- und HP-Maschinen hinweg anbieten zu können.

DEC und Microsoft beteiligen sich seit November 1993 an diesem Rennen, seitdem DEC ankündigte, man wolle OLE 2.0 nach außen wie CORBA erscheinen lassen. Diese Ankündigung ließ die Branche aufatmen, da hierdurch die Anti-CORBA-Position von Microsoft aufgeweicht wurde.

Sun und NeXT bilden seit Dezember 1993 eine dritte Allianz, die aber noch viele Fra-

gen offen läßt. Ersten Zeitungsberichten zufolge wird die NeXTstep-Entwicklungsumgebung ergänzend zum DOE mit dem Betriebssystem Solaris von Sun ausgeliefert. Mitte 1994 wird der neue Standard CORBA 2.0 voraussichtlich veröffentlicht.

Object Services

Object Services sind Dienste, die von jedem Objekt bereitgestellt werden müssen. Da jedes CORBA-Interface automatisch die Operationen vom Interface des Typs „Object" erbt, ist es nicht notwendig, diese Operationen in den Interfaces der benutzerdefinierten Objekttypen aufzuführen. Wegen ihres Umfangs wurden die Object Services in zwei Gruppen unterteilt. Die erste Gruppe der Object Services, die in der ersten Hälfte dieses Jahres standardisiert wird, umfaßt die folgenden Services:

- Lifecycle-Services beschreiben das Erzeugen und Löschen von Objekten so wie ihren Umzug innerhalb der Objektwelt.
- Event Notification Services ermöglichen die gezielte Benachrichtigung von Objekten, wenn bestimmte Ereignisse eintreten. Beispiel: Wenn eine Bestellung wegen zu geringen Lagerbestandes eines Artikels liegenbleibt, kann die Bestellung den Artikel auffordern, sie beim Eintreffen der nächsten Warenlieferung zu benachrichtigen.
- Persistence Services erleichtern die dauerhafte Speicherung von Objekten.
- Naming Services erlauben die Zuordnung von Namen zu Objekten und die Rückübersetzung von Namen in Objekt-Referenzen.

Die zweite Gruppe von Object Services wird Ende 1994 standardisiert werden und umfaßt die folgenden Services, die Gegenstand eines späteren Artikels sein werden:

- Concurrency Services,
- Externalization Services,
- Relationship Services,
- Time Services,
- Transaction Services.

Zusammenfassung

Durch die Arbeit der OMG ist eine Grundlage für die Industrialisierung der Softwarebranche (Komponentensoftware) entstanden. Schon der CORBA-Standard 1.1 vom Januar 1992 ermöglicht den Entwurf von Softwarebausteinen, die nur noch minimale Abhängigkeiten voneinander aufweisen. Erste Angebote von Softwarekomponenten liegen bereits vor. Für 1994 ist ein explosionsartiges Wachsen dieses Marktes nicht auszuschließen.

Dabei ist die Einführung von Objektorientierung nützlich, aber keine Voraussetzung: Mit der CORBA-Technik können auch herkömmliche Systeme (z.B. CICS, IMS und UTM) gekapselt werden und COBOL und C gehören noch lange nicht zum alten Eisen.

Zum Glück gibt es keine rivalisierenden Standardvorschläge, die Unsicherheit erzeugen und das Innovationstempo bremsen könnten. Auch Microsoft hat seine anfängliche Anti-Haltung aufgegeben.

Zahlreiche ORB-Produkte sind bereits verfügbar bzw. angekündigt und es ist absehbar, daß im Laufe des Jahres 1994 für alle wichtigen Umgebungen ORB-Produkte erhältlich sein werden.

Neue Softwaresysteme sollten deshalb nicht mehr ohne Berücksichtigung von CORBA erstellt werden und evtl. bereits vorhandene, hauseigene Bausteinarchitekturen sollten überprüft werden, in wie weit sie bereits mit CORBA übereinstimmen.

Von der Formulierung neuer, von CORBA abweichender Hausstandards muß aus der Sicht des Autors abgeraten werden. Neue Hausstandards sollten sich an CORBA orientieren.

Literatur

[OMG90] OMG, Object Management Architecture Guide 1990,
[OMG92] OMG, Common Object Request Broker Architecture 1992,
[Rös93a] Martin Rösch, OMG Anwender und Hersteller sollen bei CORBA 2.0 mitmachen Computerwoche 3.1.1993,
[Rös93b] Martin Rösch, Object Request Broker Funktionsweise und erste Erfahrungen, unix/mail 11 (1993)