

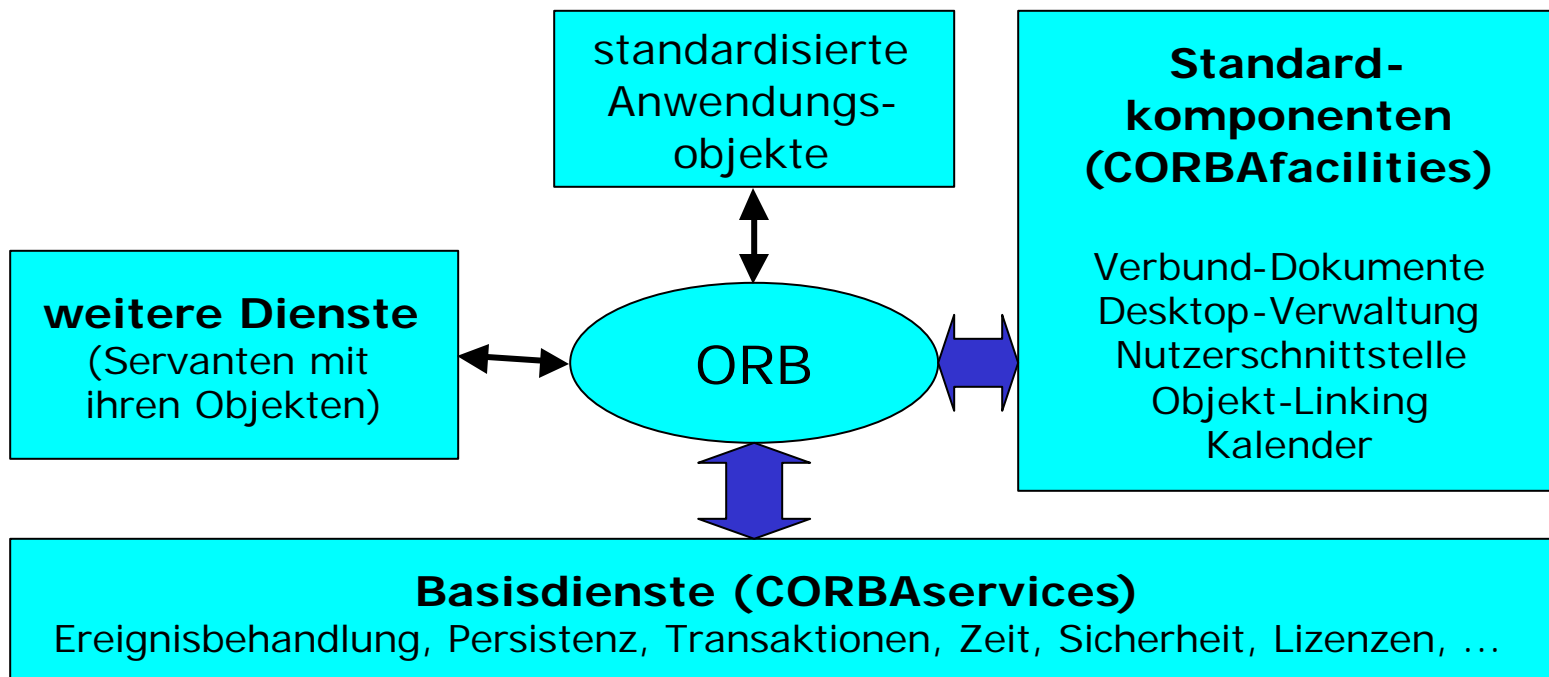
Vorlesung Software aus Komponenten

3. Komponentenmodelle

Prof. Dr. Hans-Gert Gräbe
Wintersemester 2004/05

Idee: Unterteilung der Dienste in mehr oder weniger wichtige

- Bereitstellung von Dienst-Servanten für wichtige Funktionalitäten, die in einen **Komponentenrahmen** (component framework) „eingesteckt“ werden können
 - Bsp: Geschäftsfeld-Objekte (business objects) = Objekte, die direkte Geschäftsprozessabstraktionen repräsentieren
 - steht erst ganz am Anfang der Entwicklung



Basisdienste (CORBAServices)

- Fokus auf fundamentale Bausteine **jeder** Implementierung, welche die Komponenten-Infrastruktur zur Verfügung stellen sollte
 - z.B. Ereignisbehandlung, Transaktionsabwicklung, Namensverwaltung
 - derzeit 16 Basisdienste spezifiziert
- 2 Kategorien:
 - Dienste zur Unterstützung unternehmensweiter verteilter Anwendungen
 - nutzen typischerweise CORBA-Objekte als Moduln und CORBA als Kommunikations-Middleware
 - grob granulare Ebene, CORBA als „Objektbus“
 - Dienste zur Unterstützung fein granularer verteilter Anwendungen
 - Bedeutung nimmt ab, da zu hoher Komplexitätsgrad
 - Ausnahme: persistent state service (PSS) als einer der Pfeiler des CCM
- große CORBA-Anwendungen nutzen oft nur wenige Basisdienste
 - verfügbare CORBA-Plattformen bieten deshalb oft nur Implementierungen einiger Basisdienste an

Dienste zur Unterstützung grob granularer verteilter Anwendungen

- Namensdienst (name service)
 - Abbildung intern verwendeter UUID auf externe Bezeichner
 - Namens-**Kontexte** und Kontext-**Hierarchien**
 - vergleichbar zu Verzeichnisstrukturen
- Händlerdienst (trader service)
 - Verfeinerung des Namensdiensts (white vs. yellow pages)
 - Anbieter veröffentlichen Dienstangebote per **Registrierung**
 - Nutzer finden Angebote über Händlerdienst per **Beschreibung**
 - Händlerdienste organisieren Angebote in Handels-**Kontexten**
 - Standardisierte Methoden zur Suche in den Angeboten

- Ereignisdienst (event service)
 - Verteilung der E.-**Objekte** von E.-**Erzeugern** (event supplier) an E.-**Konsumenten** (event consumer)
 - E.-Objekte sind unveränderbar, wenn einmal erzeugt
 - strikt unidirektionaler Informationsfluss
 - E.-**Kanäle** (event channel) entkoppeln Erzeuger und Konsument
 - E. können getypt sein (OMG IDL)
 - Kanäle können Ereignisse nach ihrem Typ filtern
 - Push- und Pull-Methoden werden unterstützt
- Benachrichtigungsdienst (notification service)
 - Erweiterung des Ereignisdiensts um einige kritische Merkmale
 - Dienstqualität, Administration
 - dynamische E.-Filterung, Filterung auf verschiedenen Ebenen
 - technisch kein Basisdienst, sondern Standardkomponente
 - gemeinsamer Standard mit Telecomm. Domain Task Force

- Transaktionsdienst (object transaction service, OTS)
 - einer der wichtigsten Bausteine für verteilte Anwendungen
 - standardisiert seit 1994
 - wird von den meisten ORB-Produkten und J2EE-Servern unterstützt
 - **Eingebettete Transaktionen** nur optional
 - Transaktionshülle um Folge von Operationen
 - erforderlich zur unabhängigen Entwicklung auf verschiedenen Hierarchie-Ebenen
 - (noch) nicht standardisiert, weil kaum eines der heute ex. Transaktionssysteme so etwas vorsieht
 - Verwaltung eines (objektspez.) aktuellen Tr.-**kontexts** durch OTS
 - Objekte müssen dazu die Schnittstelle *TransactionalObject* implementieren
 - Methoden *begin*, *commit*, *rollback* operieren auf dem Kontext
 - Objekte unter Transaktionskontrolle registrieren sich beim OTS-Koordinator-Objekt

- Transaktionsdienst (Fortsetzung)
 - Ressourcen müssen die Schnittstelle *Resources* implementieren
 - Koordinator wickelt darüber 2-Phasen-commit-Protokoll ab
 - bekanntes Problem der Deadlock-Gefahr
 - 3-Phasen-Protokoll vermeidet diese, ist aber teurer
 - heute weit verbreitet: Transaktionskontrolle nicht als separater Dienst, sondern als Kontextkontrolle **innerhalb** eines Anwendungsservers
 - Diese Abstraktion wird vom CCM abgedeckt
- Sicherheitsdienst (security service)
 - erforderlich, wenn sich verteilte Anwendung über mehrere Vertrauensbereiche (trusted domains) erstreckt
 - spezifiziert in **CORBAsecurity**
 - Authentifizierung, sichere Kommunikation, Zertifizierung
 - volles Spektrum wird derzeit von kaum einem Produkt unterstützt
 - meist nur SSL-basierte Sicherheit
 - unterstützt einfache Sicherheit, aber keine Zertifikate

Dienste zur Unterstützung fein granularer verteilter Anwendungen

- Nebenläufigkeits-Kontrolldienst (concurrency control service)
 - Synchronisierung nebenläufiger Zugriffe auf Ressourcen
 - *read* (nicht exklusiv lesen), *upgrade* (exklusiv lesen), *write*
 - Geschützte Ressourcen verwalten dazu Schlossmengen (lock sets), die von einem Koordinator-Objekt erzeugt und verteilt werden
- Lizenzdienst (licensing service)
 - Verwaltung von Objektlizenzen, Abrechnung von Gebrauchsgebühren für Objekte
 - Unterstützung verschiedener Lizenzmodelle
 - 2 Schnittstellen: *Lizenzdienst-Manager* (LDM) und *Lizenzdienst*
 - Objekt unter Lizenz (OL) erfährt über LDM, unter welchen Bedingungen seine Nutzung legitimiert ist
 - OL fordert vom LDM Referenz auf entsprechendes (hersteller-spezifisches) Lizenzdienst-Objekt (LDO) an

- Lizenzdienst (Fortsetzung)
 - OL informiert LDO über **Kontext** der Lizenzanforderung
 - LDO prüft, welche Nutzung des OL in dem Kontext legitim ist
 - LDO veranlasst Übergang von OL in erlaubten Zustand (ggf. Demo-Modus, Probe-Modus)
 - OL informiert LDO über Ende der Nutzung
 - aktuelle Lizenzgestaltung also gekapselt zwischen OL und LDO
 - zwischen beiden kann auch statistisch relevante Information ausgetauscht werden
 - Nutzerprofile, Lizenzdauer und –ablauftermine
- Zeitdienst (time service)
 - Synchronisierung der Uhren in verteilten Systemen
 - Korrelation innerhalb sinnvoller Fehlerschranken, um zeitliche Kausalitäten über Systemgrenzen hinweg zu erhalten

- Lebenszyklusdienst (lifecycle service)
 - Verwaltung von Objekten (Erzeugen, Kopieren, Löschen, Verschieben) oder Gruppen von Objekten
 - unterstützt Objekterzeugung durch Factory-Objekte
 - Registrierung, Wiederverwendung letzterer
 - Objektverwaltung mit Referenzzählern in verteilten Anwendungen oder mit verteiltem garbage collection wird nicht unterstützt
 - Grund: verteiltes garbage collection in fehleranfälliger Umgebung (Maschinen- oder Netzwerkausfall) ist sehr kompliziert, braucht Transaktionskontext
 - kein Problem beim Einsatz von CORBA als Kommunikations-Middleware, da dort Objekte gewöhnlich Serverobjekte mit unbegrenzter Lebensdauer
- Beziehungsdienst (relationship service)
 - Erzeugen, Löschen und Verwalten von Beziehungen zwischen Objekten, Navigation über Beziehungen

- Persistenz-Dienst (persistent state service, PPS)
 - Persistenz = Eigenschaft eines Objekts, das Programmende zu überleben
 - CORBA 2: Persistenzobjekt-Dienst (persistence object service, POS)
 - seit 1994, erste Implementierungen 1996
 - unterspezifiziert: konkrete Speichieranforderung war anwendungsspezifisch gelöst
 - CORBA 3: Ablösung durch Persistenzzustands-Dienst
 - Grundlegender Ansatz: Trennung von persistentem Objekt und Persistenzmechanismus
 - Dateien, Datenbanken
 - strukturierte Speicher (Containerdokumente)
 - sehr einfache Schnittstelle: Speichern und Laden eines Objekts
 - drei problematische Objekteigenschaften:
 1. Objekte haben Identität, sind nicht referenziell transparent
 - Problem beim mehrfachen Speichern / Laden

- Persistenzzustands-Dienst (Fortsetzung)
 - 2. Objekte können sich aufeinander beziehen (Objekt-Web)
 - Beziehungen müssen mit gespeichert werden
 - wesentliche und flüchtige Beziehungen
 - Probleme beim Mehrfachspeichern (RAID etc.)
 - 3. Objekte sind Einheiten der Datenkapselung
 - Sicherung der Integrität von Objekten auf dem Speichermedium
 - Schutz vor Manipulation unter Umgehung der Objekt-Schnittstelle
- POS löste Probleme durch Kooperation zwischen Objekt und Persistenzdienst über ein Protokoll
- PSS: explizite Deklaration, welche Objektteile wie zu speichern sind
- neue OMG Beschreibungssprache für solche Deklarationen (**persistent state description language, PSDL**)
 - Spezifikation verschiedener abstrakter und konkreter Speichertypen (analog Schnittstellen und Klassen in Java)
 - Spezifikation entsprechender Factories

- Auslagerungsdienst (externalization service)
 - Linearisierung / Delinearisierung von Objekten
 - zueinander invers (erzeugt Objektkopie)
 - keine referenzielle Integrität
 - Wertkopie von Teilobjekten
 - Referenzen **nur** über ORB Referenzmechanismus
 - zum Datenexport von Objekten in Dateien und Streams
 - Schnittstelle *Streamable* des auszulagernden Objekts (AO)
 - wird von Strom-Objekt gerufen, das selbst Schnittstelle *Stream* implementiert
 - über *externalize_to_stream* Methode des AO
 - erzeugt daraus ein lineares Objekt (LO), das Schnittstelle *StreamIO* implementiert
 - es können ganze Graphen von Objekten ausgelagert werden.

- Eigenschaftsdienst (properties service)
 - dynamisches Binden von Eigenschaften an Objekte
 - keine semantische Interpretation der Eigenschaften
 - Schnittstelle *PropertySet* mit Methoden *add*, *modify*, *delete*
 - diese können normal, read-only (löscher, schreibgeschützt), fixed-normal (nicht löscher) oder fixed-read-only sein
- Anfragedienst (object query service)
 - Dienst zum Auffinden von Objekten nach Attributen
 - ähnlich Händlerdienst, sucht aber Objektinstanzen
 - Unterstützt Object Query Language (OQL-93) der Object Database Management Group) sowie SQL mit Objekterweiterungen
 - Definiert Schnittstelle eigener *Sammeldienst*-Objekte
 - Semantik geordneter Mengen (*add*, *remove*, *enumerate*)
 - spezielle Schnittstelle *Iterator* zur Auswertung solcher Objekte
 - Anfrage-Objekt kapselt die Anfrage, welche in zwei Schritten beantwortet wird: Vorbereitung und Abarbeitung der Anfrage

- Anfragedienst (Fortsetzung)
 - Vier Objekttypen:
 - Anfrage-Objekte (query object, QO) und Sammelanfragen (querable collections, QC)
 - Anfrage-Auswerter (query evaluator, QE) wertet QO oder QE aus und erzeugt Ergebnis-Sammelobjekt
 - Anfrage-Manager (query manager, QM) erzeugt QO oder QE und schickt sie an QE zur Beantwortung
 - Das Objekt, das Anfrage generiert, benutzt *Iterator*-Schnittstelle zur Auswertung der Antwort
- Sammeldienst (object collections service)
 - Möglichkeit zum Bilden von Sammeltypen verschiedener Topologien, z.B. Mengen (bags, sets), Schlangen (queues), Listen (lists) oder Bäume (trees), entsprechend der Smalltalk-Klassifikation
 - unklar, ob das nicht lieber auf Objektebene realisiert sein sollte
 - existieren effiziente Implementierungen dieser Datentypen auf Bibliotheksebene

Standardkomponenten (CORBAfacilities)

- Standardisierung von häufig benötigten Anwendungsbestandteilen
 - Komponentenrahmen zur einfachen Integration von Anbieterlösungen
- Abgrenzung von Bereichen horizontal oder vertikal
 - horizontal: Fokus auf generellem Anwendungsmodell
 - Standards für Nutzerschnittstellen, System- und Aufgabenverwaltung
 - vertikal: Focus auf bereichsspezifischen Einsatzfeldern
 - im Rahmen von OMG SIG's oder Domain Task Forces
- Standards zur Integration häufig benötigter Dienste als „Plugins“ in bestehende Komponentenrahmen (component frameworks)
 - vereinfacht und standardisiert das Vorgehen bei der Integration von Komponenten verschiedener Anwender
- Einteilung der Rahmen nach horizontalen (allgemeinen) oder vertikalen (bereichsspezifischen) Gesichtspunkten

Horizontale (generale) Standardkomponenten

- Fokus auf generellem Anwendungsmodell
 - Standards für Nutzerschnittstellen, System- und Aufgabenverwaltung
- OMG hatte hier ursprünglich folgende Rahmen im Auge
 - **Benutzerschnittstelle** (user interface)
 - **Informationsverwaltung** (information management)
 - **Systemverwaltung** (system management)
 - **Aufgabenverwaltung** (task management)
- wird heute nur noch wenig vorangetrieben und stärker auf übergreifende Dienste konzentriert, die aus branchenspezifischen Standards herrühren
 - Internationalisierung, mobile Agenten, Zeit- und Druckdienst-Standards
 - keine Standard-Komponenten, sondern Komponenten-Standards
- Standardisierungsbemühungen der ursprünglichen Bereiche spielen praktisch so gut wie keine Rolle

Vertikale Standardkomponenten

- ursprünglich Fokus auf Basisfunktionalität für unterschiedliche Marktsegmente
- Ergebnisse bekommen zunehmend segmentüberschreitende Bedeutung
 - Komponenten-Standards statt Standardkomponenten
- Ausgehandelt in Aktivitäten verschiedener Domain Task Forces
 - business enterprise integration
 - command, control, communications
 - Finanzbereich
 - Bereich Gesundheitsvorsorge
 - Lebenswissenschaften
 - Produktionsstrukturen
 - Telekommunikation usw.