

Anbindung von Legacy-Software an Enterprise-Architekturen

Sascha Hunold

`hunold@uni-bayreuth.de`

Universität Bayreuth

Dagstuhl, 15. Februar 2007

TransBS

Enterprise Applications

Charakteristiken

- Stehen im **Gegensatz** zu Desktop-, System-, wissenschaftl. Anwendungen
- **Eng integriert** in **interne** Organisation und Geschäftsmodell
- **Abhängigkeiten** zu anderen Abteilungen, externen Partnern
- wenig komplizierter Code
COBOL-Programm für Abrechnung / Marsroboter mit eingebetteter (multithreaded) Software)
- **Daten** (business data) haben **lange Lebenszeit**
- **Konstantes Anpassen** an neue Anforderungen

Legacy Software

Klassische Legacy Software

- **monolithische** Architektur
- **keine** klare **Trennung** von Präsentations-, Geschäftslogik-, und Datenschicht
- Prozedurale Sprachen wie COBOL, Fortran, PL/1, Assembler
- **Keine Dokumentation**
- **Kein Quellcode** für einzelne Programmteile
- entwickelt von **hundert**en von **Entwicklern** über mehrere Jahre

Transformation der Legacy-Software

Konzepte

- **Analyse** und **Dokumentation**
- Auswahl der EAI-Lösungen
- Verbesserung auf Programmebene und Systemebene
- **Sprachmigration**
- **Konsolidierung** der Daten
- Aktualisierung des **Front-Ends**
- Neugestaltung Geschäftslogik
- Verpacken des hybriden Softwaresystems (neu/Legacy)

Moderne Softwarearchitekturen

Typische Struktur moderner Software

- **Präsentations- und Benutzerschicht**
 - Benutzerservices wie Eingabe, Dialoge, Anzeigeverwaltung
- **Geschäftslogik**
 - bietet Dienste der Geschäftslogik, die zwischen Anwendungen gemeinsam benutzt werden
- **Datenschicht**
 - enthält das DB-Management und verwandte Funktionen

Auswahl einer Plattform

Enterprise Plattformen

- Architektur multi-tiered
- Komponenten-basierend
- Enterprise Services
 - Transaktionen
 - Sicherheit
 - Persistenz


 relativ wenig Auswahl

- J2EE (Sun)
- CORBA (OMG)
- .NET (Microsoft)

Auswahl einer Plattform

Enterprise Plattformen

- Architektur multi-tiered
- Komponenten-basierend
- Enterprise Services
 - Transaktionen
 - Sicherheit
 - Persistenz

 relativ wenig Auswahl

- J2EE (Sun)
- CORBA (OMG)
- .NET (Microsoft)

Java Enterprise Architektur

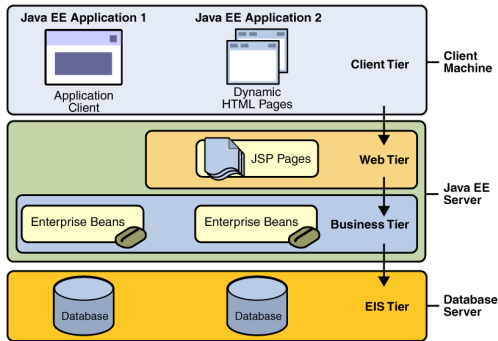
Eigenschaften

- Basiert auf Java-Technologie ➡ unabhängig von Plattform (aber limitierend)
- Kontrolliert vom JCP (Java Community Process)
- J2EE – Menge an Spezifikationen, Frameworks, Technologien
- Verteilte Business Anwendungen
- Umfasst Sprachen: Java, XML, SQL, JSP, HTML

Multi-tiered Java Enterprise Anwendung

Typische Business-Anwendung

Client Tier / Middle Tier / Data Tier



Java EE Server

Open Source

- JBoss
- JOnAS
- GlassFish
- Apache Geronimo

Proprietär


- IBM WebSphere
- BEA WebLogic
- SAP Web Application Server
- Oracle Application Server

Kommunikation zwischen Komponenten

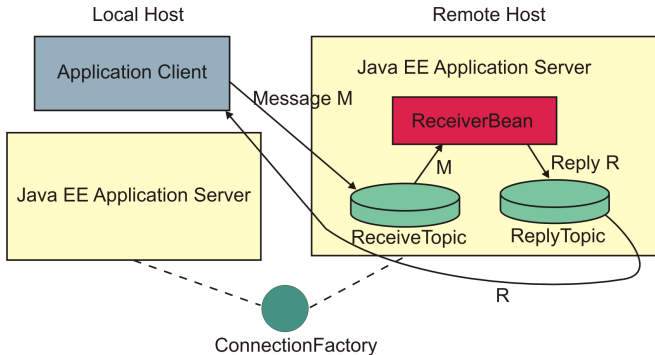
- HTTP(S)
 - Transfer von Markup-Seiten
 - Client - Web Component Tier
- RMI-IIOP
 - Erweiterung von RMI um Inter-Orb Protokoll
 - Kommunikation zwischen CORBA und Java-Komponenten
- JMS
 - Kommunikation zwischen Tiers
 - MOM (message-oriented middleware)
 - MOM benutzt für Inter-Applikationskommunikation (JMS kann diese wiederverwenden und erweitern)
- JDBC
 - Einheitliches Interface für versch. relationale DBMS

JMS (Java Message Service)

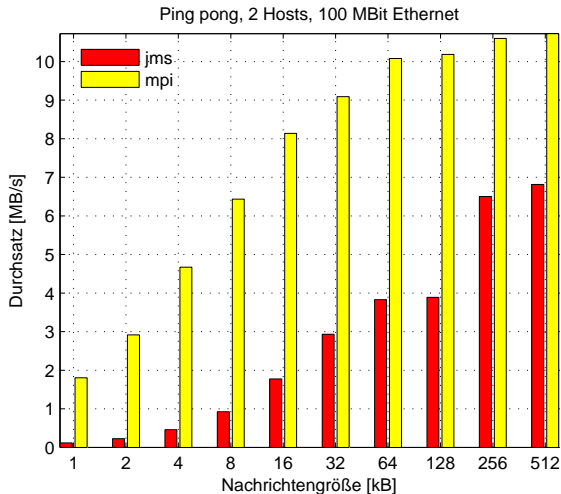
Eigenschaften

- Bestandteil von J2EE, **asynchron** (Message Broker)
- **Entkopplung** (keine Referenzen)  erleichtert Austausch einzelner Systeme
- **Flexibel** konfigurierbare Kommunikation (Notebooks)
- **Unabhängig** von Plattform, Sprache
- **Verteilte** Anwendungen, Transaktionen
- Standardisierung
- Definierte Sicherheitsmechanismen
- Skalierbarkeit

Leistungsanalyse von JMS



Leistungsanalyse von JMS



CORBA

- Common Request Broker Architecture
- beinhaltet Spezifikationen für
 - Programmiersprachen
 - Plattformen
 - betriebssystemunabhängiges verteiltes Objektmodell
- entwickelt von **Object Management Group (OMG)**
(erste Version 1991)
- J2EE unterstützt CORBA 🖱️ wichtige Integrationsmöglichkeit

Wie erzeugt/nutzt man ein verteiltes Objekt?

- Interface Definition Language (IDL)

```
interface Account {  
    void deposit(in unsigned long amount);  
    long balance();  
}
```

- *IDL-Compiler*: Server-Skeleton und Client-Stub
- *Server*: implementiert Skeleton, erzeugt Instanz
- *Client*: Instanz des Stubs, Methodenaufruf auf Stub
- *ORB*: (1) in-Argumente → Server; (2) Methodenaufruf; (3) out-Argumente → Client

ORB-Implementierungen

- frei

- ILU (CORBA 2.0)
- **JacORB** (CORBA 2.3, Java)
- MICO (CORBA 2.3, C++)
- omniORB (CORBA 2.6, C++)
- OpenORB (CORBA 2.4.2, Java)
- ORBit2 (CORBA 2.4, C, C++)
- TAO (CORBA 2.4, C++)
- **SunORB**, Java IDL (Sun, CORBA 2.3.1, Java)
- **MTDORB** (CORBA 2.6, Delphi)

- kommerziell

- Orbix (CORBA 2.6, C++)
- Orbacus (CORBA 2.6, C++, Java)
- **Visibroker** (CORBA 2.6, Java, C++, .NET)
- Tuxedo (BEA)

Und was ist mit TransBS?

Was wir haben...

GBware

- monolithische Anwendung
- Delphi (*Fat Client*)
- DCOM

Was wir wollen...

- verteilt, heterogen
- Unabhängigkeit von Sprache (IDE)
- Modularität (Erweiterbarkeit, Thin Clients, Web-Interface)

Transformation von GBware

Strategie

- **Datenkonsolidierung** (DB-Schema)
- Extraktion der Services
- **Modellierung** der **Services** (mit Werkzeugen zur Modellgetriebenen Entwicklung)
- Verifikation mit **Unit-Tests**
- Geschäftslogik zusammenführen
- Code-Wrapping (in Services?)

Wohin mit der Geschäftslogik?

Übliche Aufteilung

35 %



Client Application

0 %



Business Layer

65 %



Database

Wohin mit der Geschäftslogik?

Optimale Aufteilung

0 %



Client Application

100 %



Business Layer

0 %



Database

Wohin mit der Geschäftslogik?

Realistische Aufteilung

3 %



Client Application

100 %



Business Layer

0.5 %

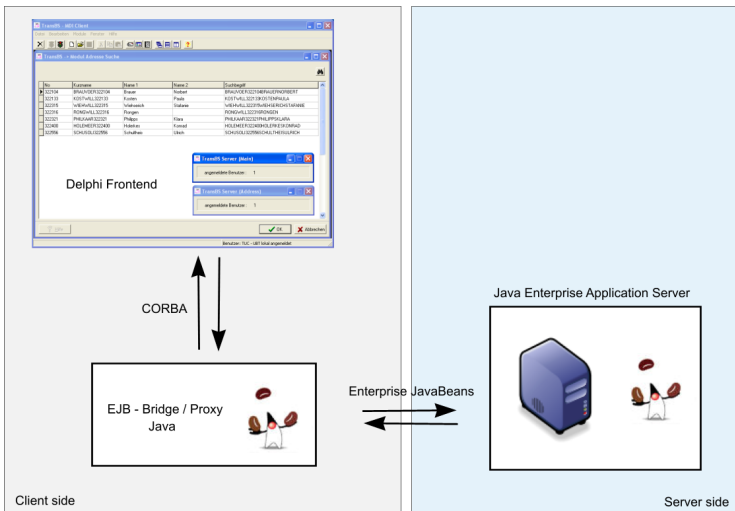


Database

Probleme mit EAI

- kein Delphi-CORBA-Standard der OMG
- wenige freie ORB-Implementierungen (wenig Dokumentation)
- Inkompatibilitäten zwischen ORBs

Vorschlag



CORBA-Java-Integration

- Versuch Java und Delphi über CORBA zu verbinden
- Java-Server, JacORB, GPL
- Delphi-Client, Borland Visibroker
- SunORB, CORBA, Nameserver

CORBA-Java-Integration (2)

The screenshot displays a Windows desktop environment with several windows and annotations:

- Client-Konsole:** A red arrow points to a command prompt window titled "Eingabeaufforderung" showing the execution of the `BankClient` application. The command used is `BankClient -ORBInitRef NameService=iioploc://localhost:900/NameService`.
- Nameserver:** A red arrow points to another command prompt window titled "Eingabeaufforderung" showing the execution of the `orbd` command: `C:\Programme\Java\jdk1.5.0_10\bin>orbd`.
- Delphi-Client:** A red arrow points to a Delphi application window titled "CorbaTest". The window contains a text input field labeled "Server Float" with the value "\$68.02" and a button labeled "Hole Float".
- Java-CORBA-Server:** A red arrow points to a Java console window titled "Eingabeaufforderung" showing the output of the `orbd` command, including messages like "Opened new server-side TCP/IP transport to 127.0.0.1:1101" and "Opening new account: FooBar".

Fragen

Anregungen erwünscht!