

Framework oder Application Server Gegensatz oder Ergänzung?



Application Server oder Framework - Gegensatz oder Ergänzung?

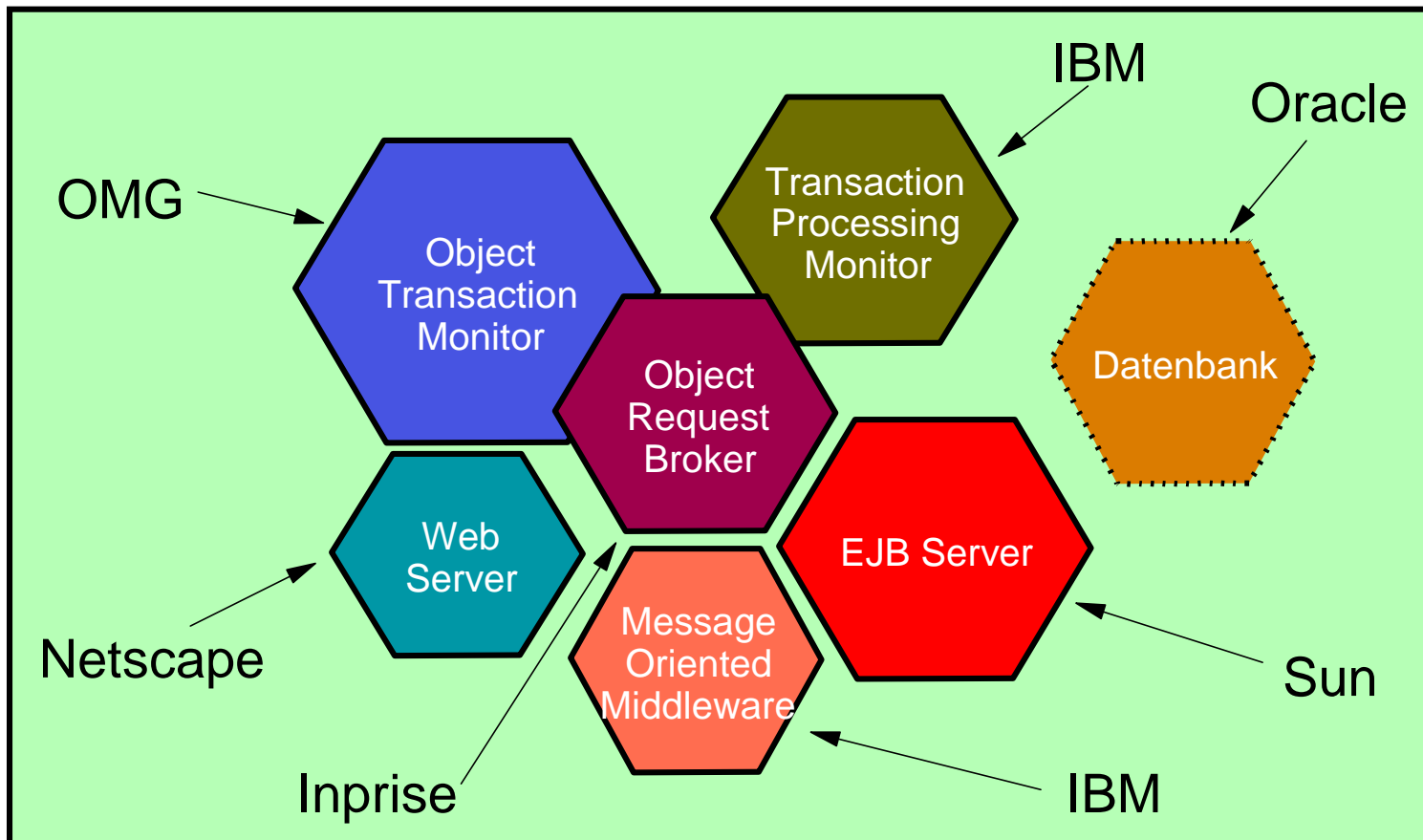
Dr. Gerhard Wanner
Email: gwanner@ibl.de

- Was ist der Leistungsumfang typischer Application Server (AS)?
- Beispiele für Frameworks (FW) für den Einsatz mit AS
- Nutzen eines Frameworks im Rahmen eines AS
- Integration von Framework und AS
- Wie hilft ein generativer Entwicklungsprozess bei dieser Integration?

Application Server

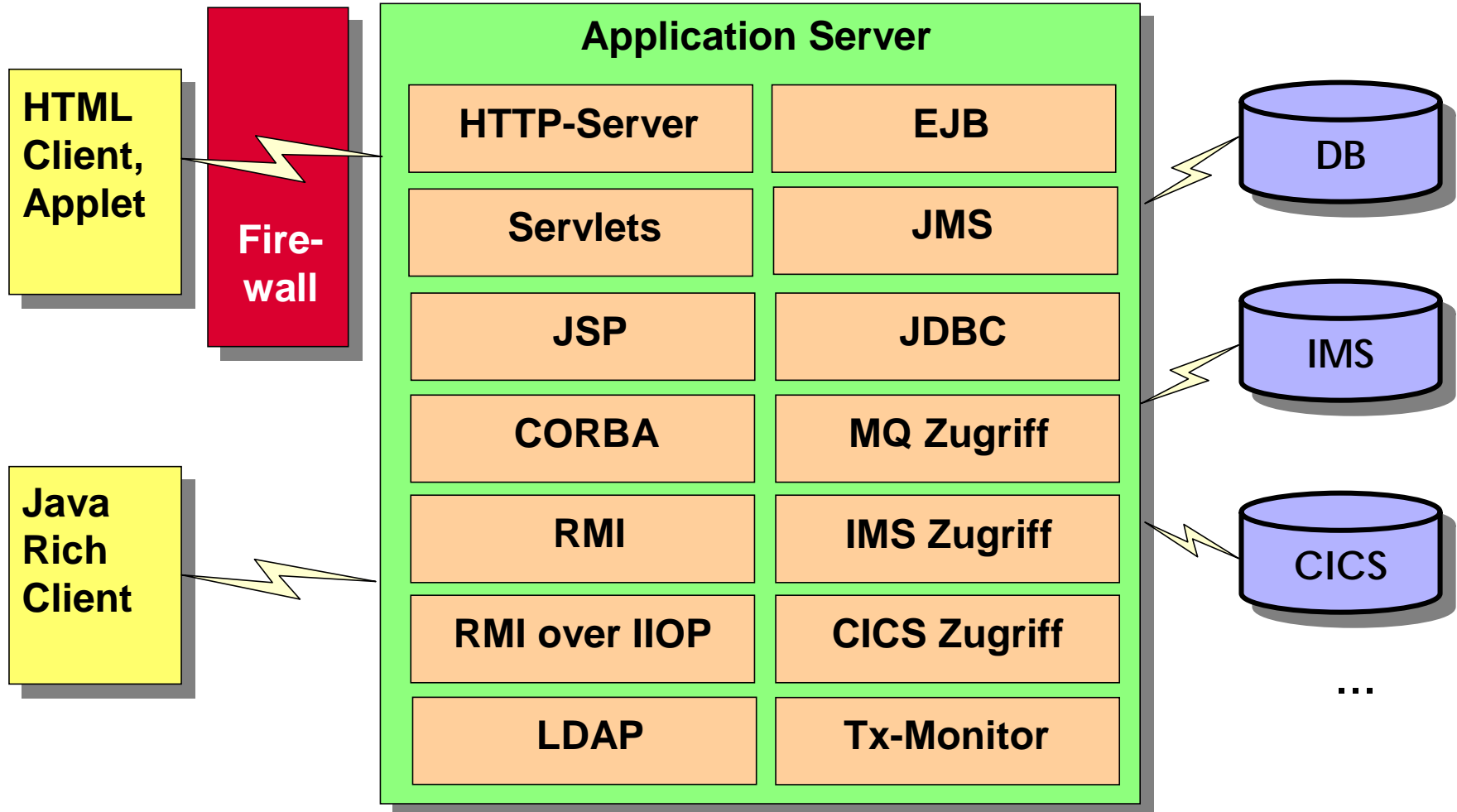
Was wird darunter verstanden?

■ Zusammenwachsen von Nischen-Technologien



- Rahmen für die Verarbeitung von Business-Logik
- Verbindung zu Back-End-Systemen
 - Datenbanken
 - Legacy-Systeme
 - Transaktionsmonitore
- Verbindung zu Front-Ends
 - Web Clients
 - Rich Clients
- Ziel: Modulares, hoch-skalierbares und robustes System
- Trotzdem dynamisch genug, um die sich schnell ändernden Business-Anforderungen zu erfüllen

Application Server Aufbau / Bestandteile



- Ein Framework ist ein durch den Softwareentwickler anpassbares oder erweiterbares System kooperierender Klassen.
In der Regel werden abstrakte oder leere Operationen in Unterklassen definiert bzw. implementiert.

Quelle: Helmut Balzert - Lehrbuch der Software-Technik
Band 1: Software-Entwicklung

- Es ist optimal geeignet, um komplexe Sachverhalte zu abstrahieren und anzubieten
- Es bietet eine vorgegebene Infrastruktur und Hilfe bei der Programmarchitektur

- Einige Leistungsmerkmale eines AS stehen als Framework zur Verfügung. Beispiele:
- Anbindung von Ressourcen an den Java Transaction Service (JTS)
 - Beispiel folgt...
- Integration von Datenquellen zur Nutzung des Connection-Poolings
 - Mechanismus der AS zum performanten Zugriff auf Datenquellen
 - Für Standard-Zugriffe (z.B. JDBC) wird das Connection-Pooling bereits vom AS bereitgestellt
 - Für andere, parallel zugegriffene, Datenquellen kann das Connection-Pooling der AS durch Verwendung des Frameworks genutzt werden

Java Transaction Service (JTS)

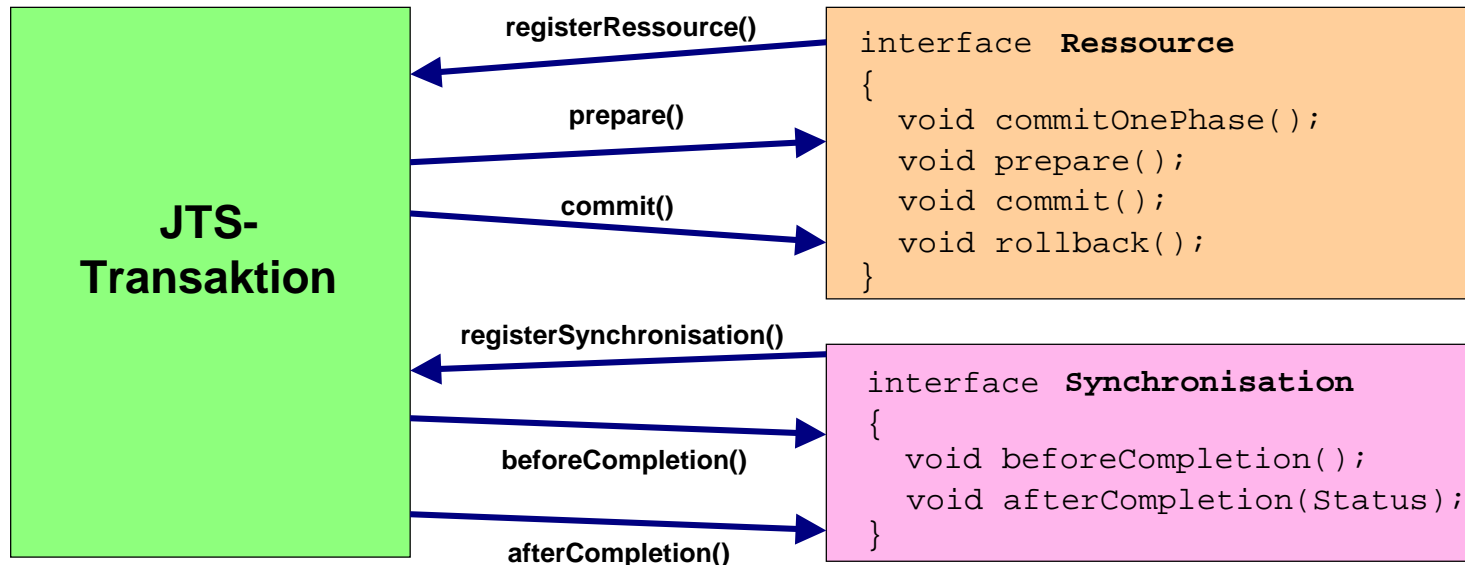
Nutzung des Frameworks

■ Ressource

- Die Ressource-Schnittstelle beschreibt Objekte, auf welche ein 2-Phase-Commit angewendet werden kann

■ Synchronisation

- Die Synchronisation-Schnittstelle erlaubt Vor- und Nach-Verarbeitungen für die an der Transaktion beteiligten Objekte



■ Persistenz-Framework

- Abbildung der Business-Objekte auf die Datenhaltungssysteme
- Detailliertes Beispiel folgt...

■ Framework für Vererbungen und Beziehungen

- Für die Erstellung einer Anwendung, die über ein Demo-Beispiel hinausgeht, sind Vererbungen und Beziehungen unverzichtbar
- Dieses Thema ist in der EJB-Spezifikation bislang ausgespart

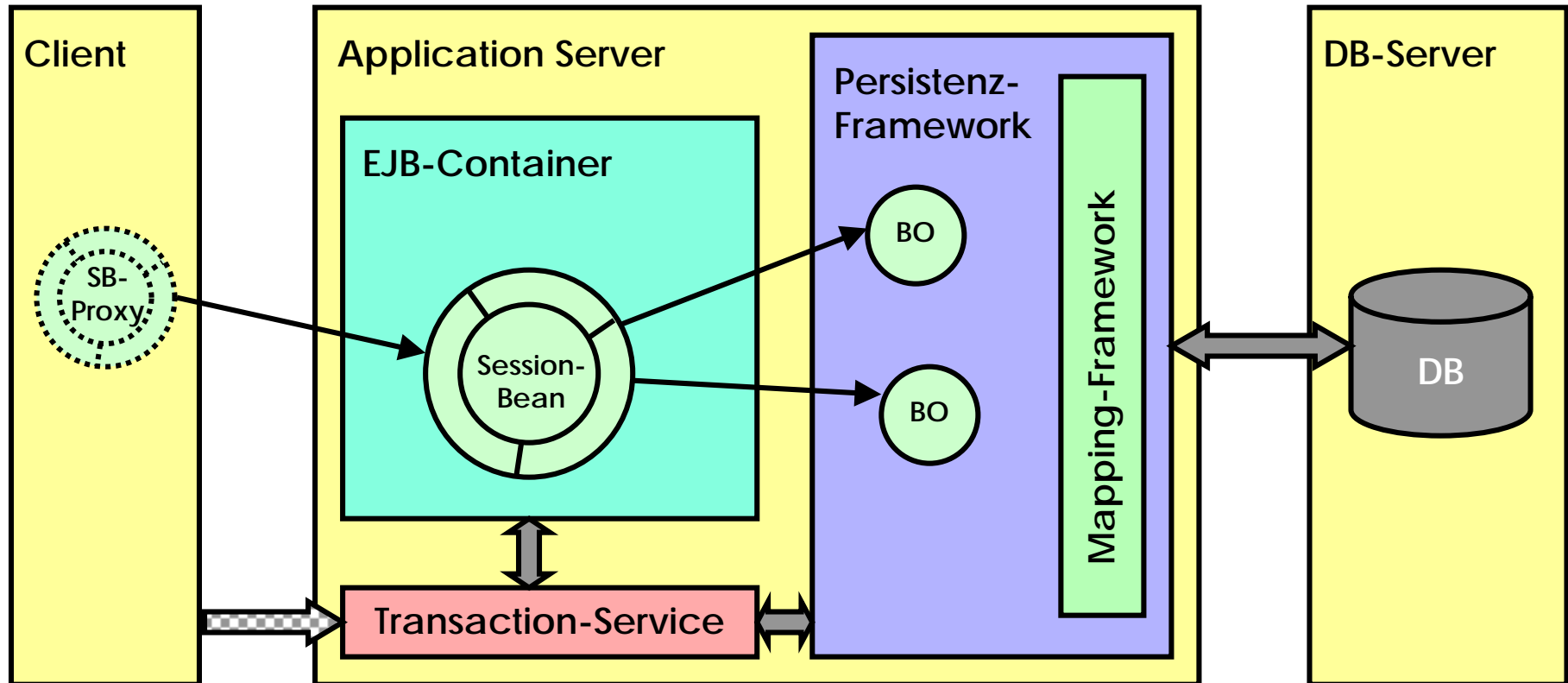
■ Framework für Exception-Behandlung

- Eine zentrale Exception-Behandlung ist notwendig
 - Fehler-Protokollierung
 - Programmiermodell sollte einfach gehalten werden
- J2EE bzw. EJB definieren keinen speziellen Service hierfür; die Probleme entsprechen daher denen anderer verteilter Systeme

- **AS enthalten (wenn überhaupt) eine rudimentäre Abbildungsmöglichkeit von Objekten auf Datenbanken**
 - Typischerweise 1:1-Abbildung zwischen Klassen und Tabellen
 - Die EJB-Spezifikation spart dieses Thema aus
- **Die AS sind aber genau dazu da, die Brücke zu allen eingesetzten Datenhaltungssystemen zu schlagen**
 - Einbindung von Legacy-Systemen
 - Einbindung verschiedenster Datenbanksysteme
- **Ein Persistenz-Framework kann diese Lücke schließen**
 - Flexible Abbildungs-Möglichkeiten
 - Entwicklung erfolgt ausschließlich mit dem vorgegebenen Programmiermodell (EJB); kein direkter Zugriff auf die angeschlossenen Systeme

Persistenz-Framework und Application Server – Integration (1)

- Session Beans greifen auf Business-Objekte zu, die nicht EJB sind



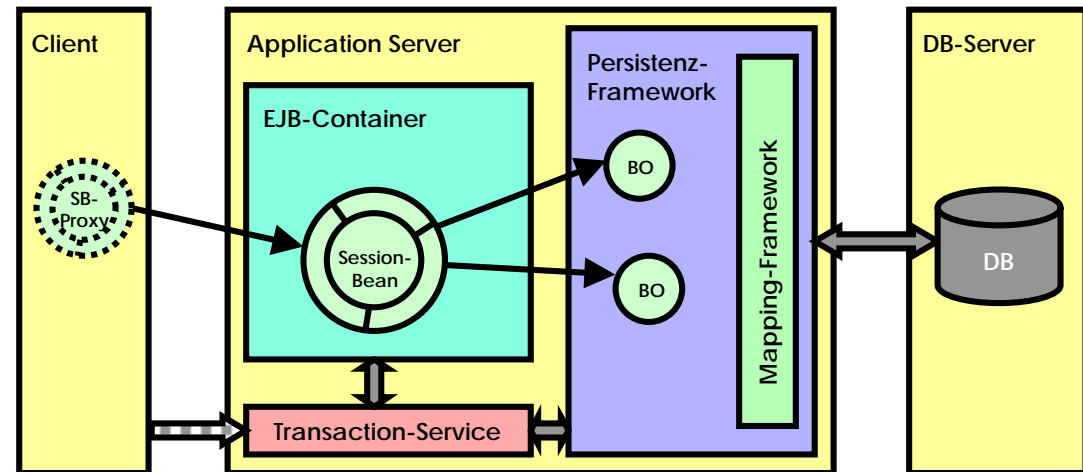
Persistenz-Framework und Application Server – Integration (1)

■ Vorteile

- Schnell, da die Business-Objekte nicht den EJB-Overhead besitzen
- Die Anzahl der Kopplungen zwischen Client und Application Server sind gering

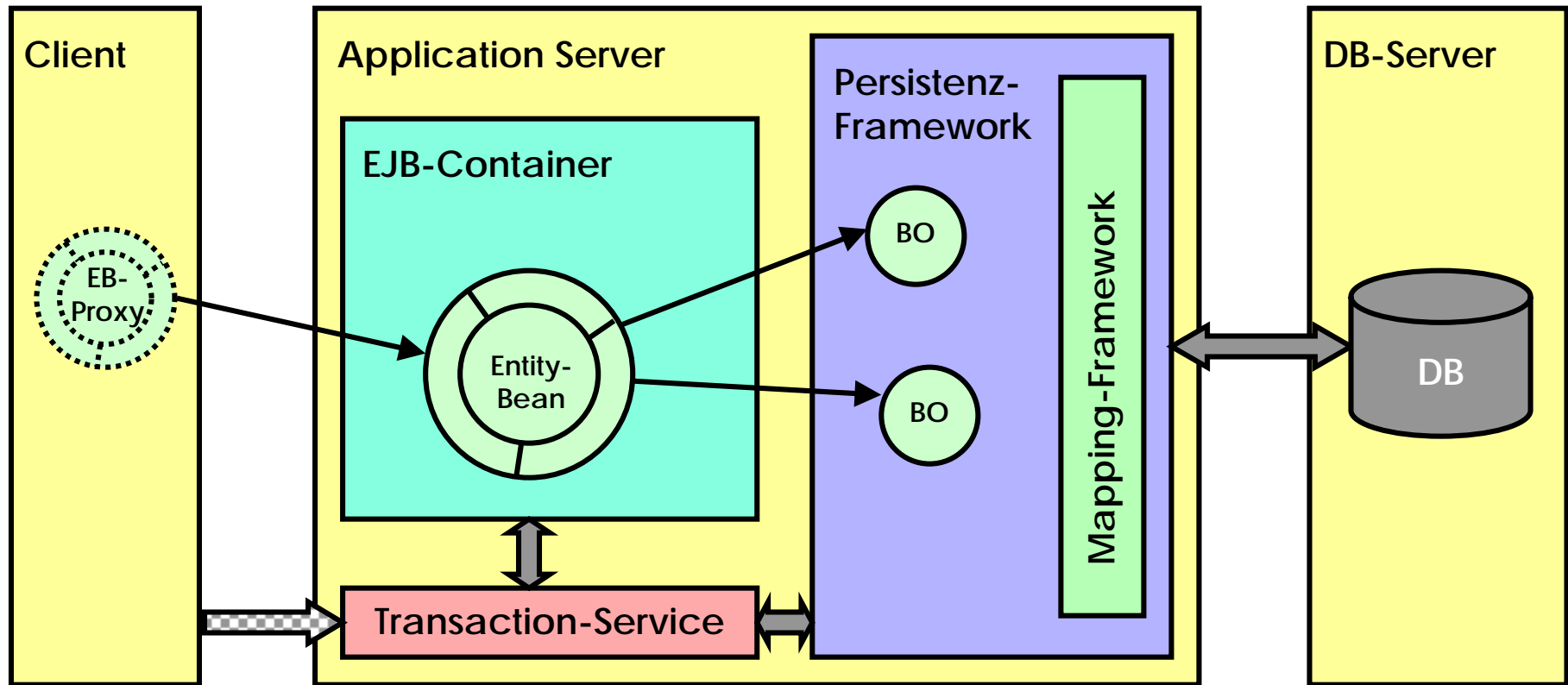
■ Nachteile

- „So war EJB wohl nicht gedacht“
- Business-Objekte können vom Client nicht angesprochen werden
- Kein deklaratives Transaktions- und Sicherheitsmanagement auf Ebene der Business-Objekte



Persistenz-Framework und Application Server – Integration (2)

- Entity Beans werden mit Bean Managed Persistence abgebildet



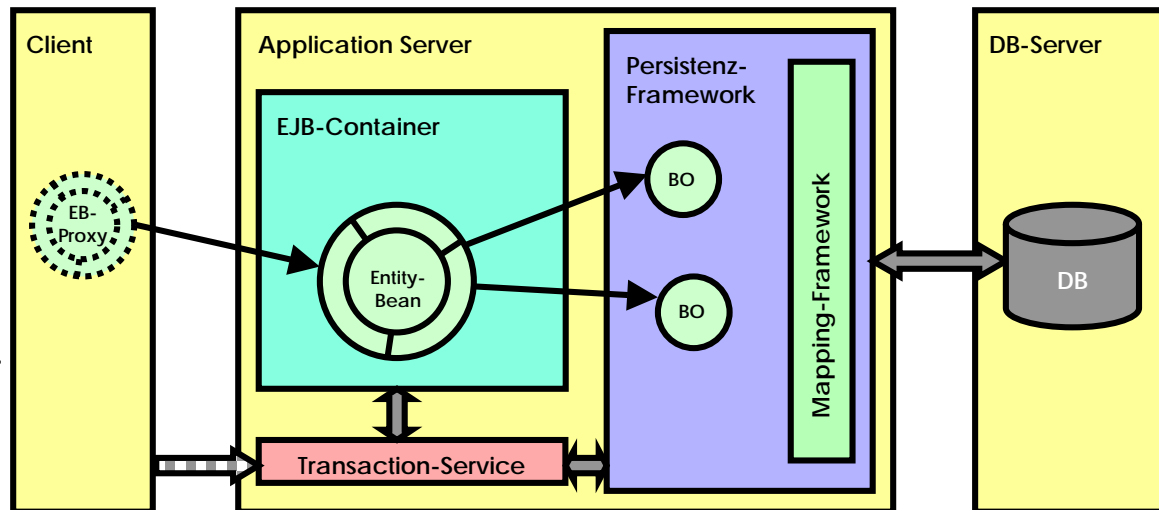
Persistenz-Framework und Application Server – Integration (2)

■ Vorteile

- Entity-Beans sind auch vom Client aus verwendbar
- Vorgehen ist konform zur EJB-Spezifikation (inkl. Transaktions- und Sicherheitsmanagement)
- Weitgehend unabhängig vom Application Server

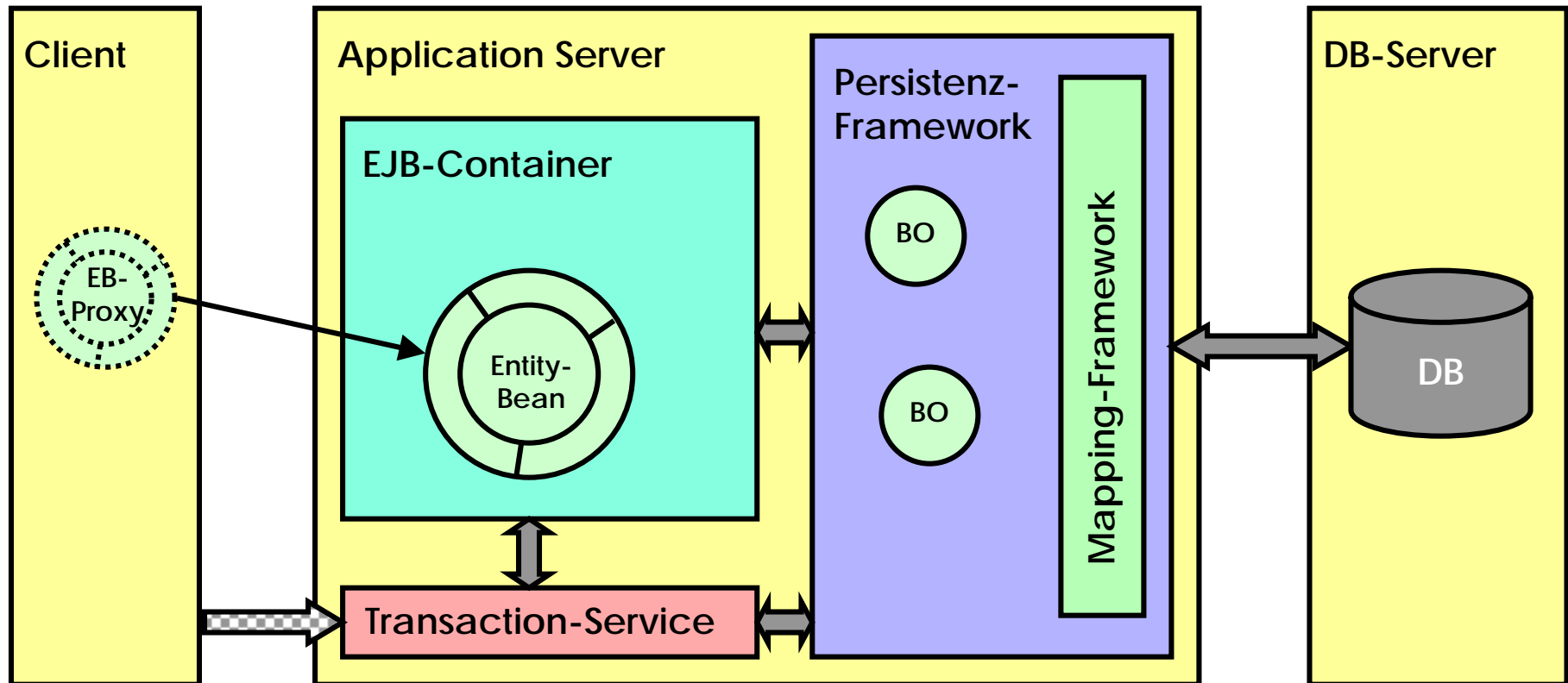
■ Nachteile

- Entity-Beans „kennen“ ihre (Persistenz liefernden) Business-Objekte
- Fertige oder zugekaufte Beans können das Persistenz-Framework nicht nutzen



Persistenz-Framework und Application Server – Integration (3)

- Entity Beans werden mit Container Managed Persistence abgebildet (Schnittstellen im EJB-Server)



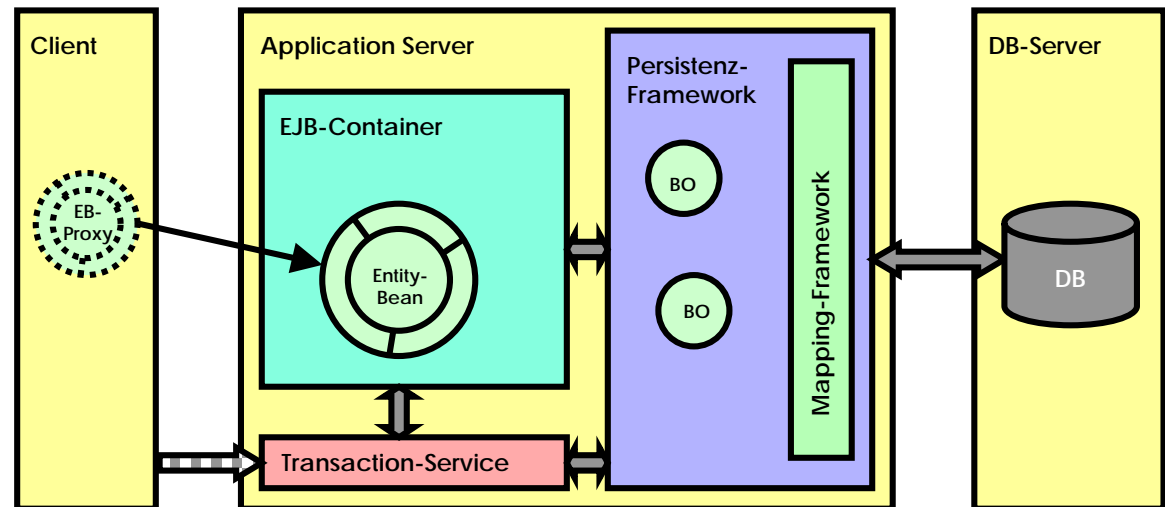
Persistenz-Framework und Application Server – Integration (3)

■ Vorteile

- Entity-Beans sind auch vom Client aus verwendbar
- Vorgehen ist konform zur EJB-Spezifikation (inkl. Transaktions- und Sicherheitsmanagement)
- Entity-Beans sind völlig unabhängig zu den (Persistenz liefernden) Business-Objekten (keine direkte Kopplung zwischen EJB-Container und Persistenz-Framework)
- Auch für fertige oder zugekaufte Entity-Beans einsetzbar

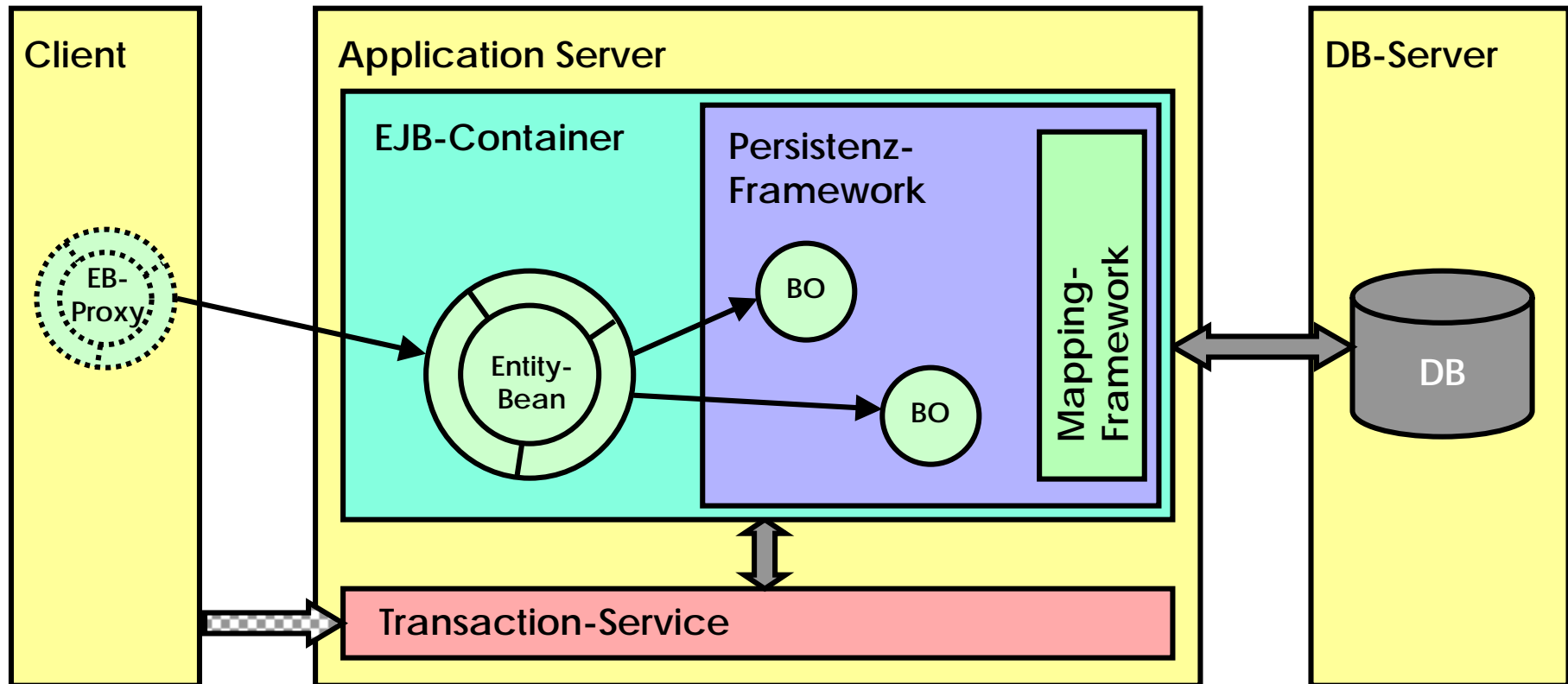
■ Nachteile

- Lösung setzt entsprechende Schnittstellen im AS voraus



Persistenz-Framework und Application Server – Integration (4)

- Entity Beans werden mit Container Managed Persistence abgebildet (eigener Container)



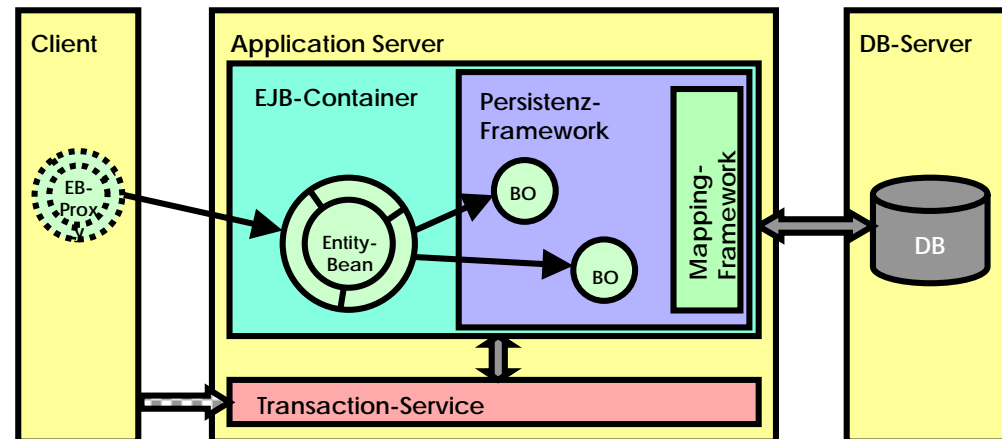
Persistenz-Framework und Application Server – Integration (4)

■ Vorteile

- Entity-Beans sind auch vom Client aus verwendbar
- Container ist potenziell für verschiedene AS nutzbar
- Auch für fertige oder zugekaufte Entity-Beans einsetzbar
- Vorgehen ist konform zur EJB-Spezifikation (inkl. Transaktions- und Sicherheitsmanagement)

■ Nachteile

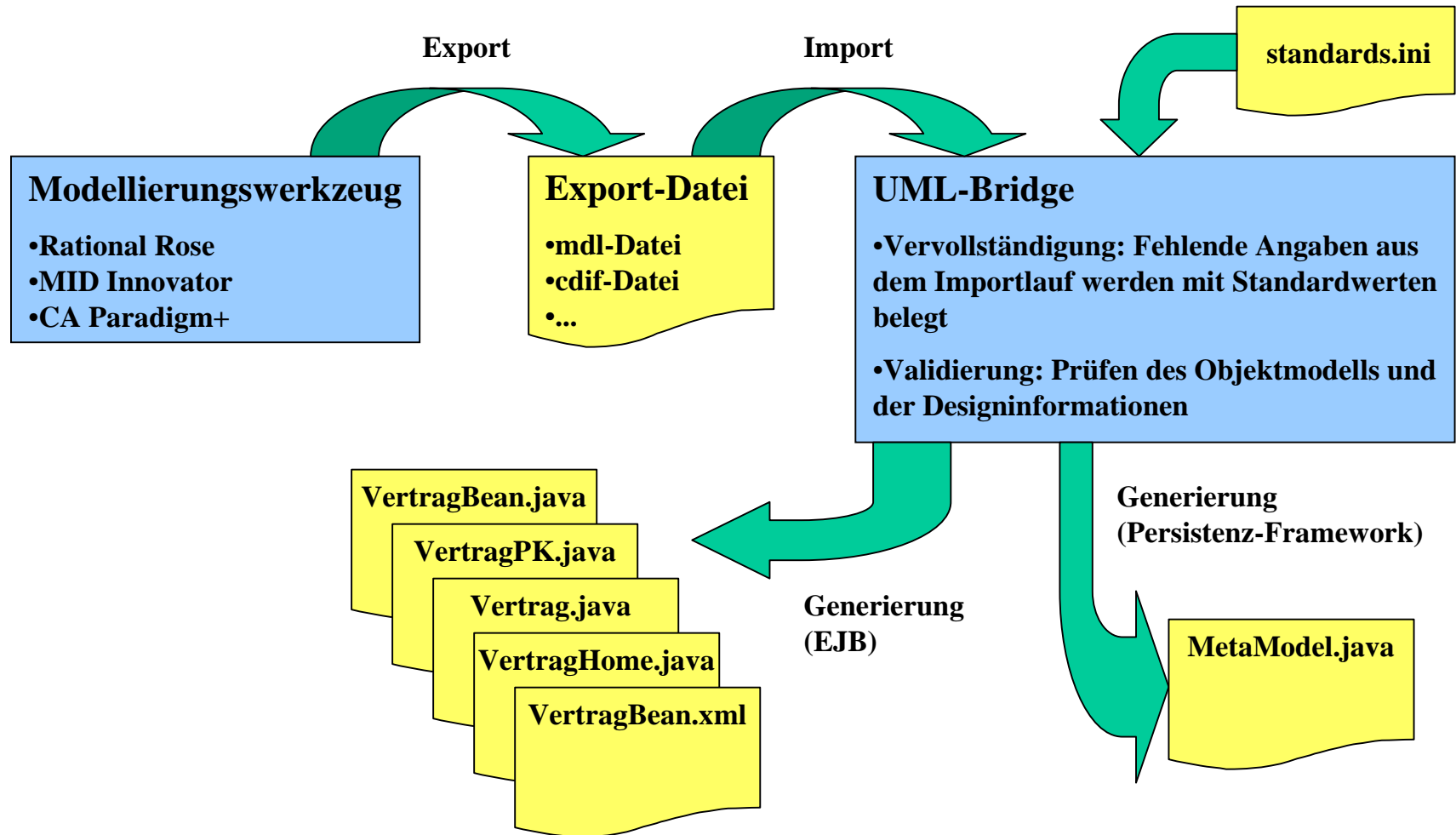
- Schnittstelle AS – Container ist nicht standardisiert, d.h. der Container kann nur für einen definierten AS realisiert werden
- Oft ist die vom Container bereitzustellende Funktionalität nicht erkennbar (kein Source)
- Nachimplementierung der Standard-Container-Funktionalität. Zusätzlich Synchronisations-Aufwände bei neuen AS-Versionen



Wie hilft ein generativer Entwicklungsprozess?

- **Welche SW-Bestandteile sind für den Betrieb eines AS notwendig?**
- **Klassen für die Business-Logik**
 - Die Business-Logik liegt in Form von Business-Objekten vor
 - Diese sind auf der Basis des ausgewählten Programmiermodells erstellt; heutzutage immer mehr in Form von Enterprise Java Beans (EJB)
- **Klassen für die Framework-Nutzung**
 - Die Nutzung der Frameworks erfolgt über die Spezialisierung der Framework-Klassen
 - Meist bestehen Abhängigkeiten zu den Business-Objekten
 - Beispiel: Persistenz-Framework
- **Die Erzeugung der notwendigen SW-Bestandteile sollte integriert erfolgen**

Generativer Entwicklungsprozess



- **Business-Objekte / Klassen zur Framework-Nutzung sind „in sync“**
 - Es gibt weniger Probleme mit unterschiedlichen Entwicklungs-Ständen
- **Der generative Entwicklungsprozess ermöglicht eine Abstraktion vom konkreten AS**
 - Aufgrund von Freiheitsgraden bei der Interpretation der EJB-Spezifikation gehen die AS-Hersteller teilweise unterschiedliche Wege
 - Ein unabhängiger Generator kann vor proprietären Details schützen
- **Unterstützung unterschiedlicher Framework-Implementierungen durch „Umschalten“ der Generierungsart möglich**
 - Beispiel: die gezeigten Implementierungsmöglichkeiten für Persistenz
- **Die Generierung ermöglicht das Füllen von Spezifikations-Lücken**
 - Beispiel: Framework für Beziehungen und Vererbung
 - Beziehungen und Vererbung werden während der Objekt-Modellierung festgelegt. Sie sollten in späteren Entwicklungsphasen nicht durch den Mangel eines Metamodells „verloren“ gehen

- Application Server sind ein wichtiger Eckpfeiler moderner IT-Architekturen
- Neben einer Vielzahl von Diensten stellen AS ein Programmiermodell, heute meist EJB, zur Verfügung
- Fehlende Dienste bzw. Funktionalitäten können durch Integration entsprechender Frameworks in den AS bereitgestellt werden
- Typische Erweiterungen betreffen Persistenz, Fehlerbehandlung und die Unterstützung objektorientierter Konstrukte
- Im Bereich Persistenz existieren verschiedene Lösungsansätze für die Integration. Nur die Bewertung der Randbedingungen ermöglicht eine Entscheidung
- Ein generativer Entwicklungsprozess unterstützt bei der Abstraktion vom konkreten AS und bei der Nutzung der Frameworks