

Techniker Krankenkasse, Hamburg: 10 Jahre produktives Java Enterprise

SEACON, 22./23.6.2009, Hamburg
Ralf Degner, Techniker Krankenkasse



Überblick

- Voraussetzungen, Ziele, Entscheidungen
- Logische Schichten
- Business-Objects (BOs): Laufzeitumgebung und Persistenz
- Das unternehmensweite OO-Modell
- Integration auf dem Client
- Erfahrungen und Erfolgsfaktoren

Voraussetzungen

- EDV in der gesetzlichen Krankenkasse (GKV)
 - Keine „normale“ Versicherung, alles ist anders
 - viele gesetzliche Regelungen, viele Anwendungen, häufige Änderungen
- Datenhaltung zentral auf dem Großrechner (DB2 / IMS-DB)
- Vorhandene Systeme:
 - 3270-Masken (COBOL)
 - 2-Schicht-Anwendungen (Centura)
- 10.500 Mitarbeiter, davon ca. 7.000 parallel online
- ca. 200 Lokationen (1 bis 350 Mitarbeiter)
- ca. 7,2 Mio. Versicherte

Ziele der Anwendungsarchitektur

Fachlich

- Optimale Unterstützung der Prozesse der TK
- Komfortable und effiziente Bedienung
- Schnelle Realisierung gesetzlicher Änderungen
- Flexible Reaktion auf neue Geschäftsmodelle (z.B. Wahltarife)

Technisch

- Moderne und langfristig tragfähige Architektur
- Gute Wartbarkeit
- Einbindung der Alt-Systeme
- Minimierung technologischer Abhängigkeiten

Entscheidungen

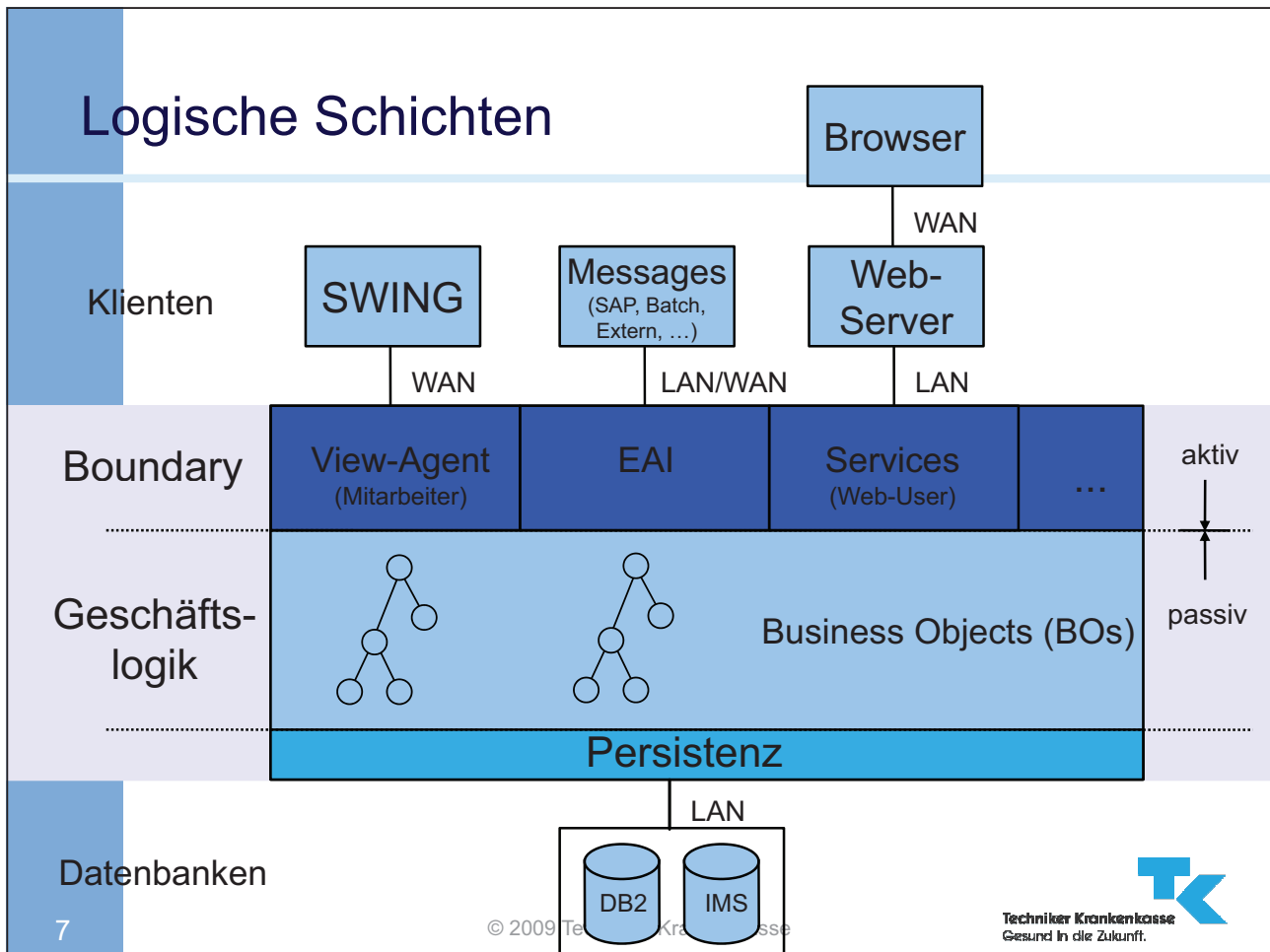
- Möglichst durchgängiger Einsatz von **objektorientierten Technologien und Methodiken**: Architektur soll dies ermöglichen
- Implementierung der Geschäftslogik im Application-Server
 - kein Masken/UPRO-Wrapping
 - Logisch „dünner“ Client
- Vollwertige GUI-Anwendung (kein HTML)
- Kein Big-Bang: schrittweise Ablösung aller Alt-Anwendungen
- Eigenentwicklung des Frameworks und Application-Servers (1998 war am Markt nichts verfügbar)
- Basistechnologien: Java und SWING

TKeasy in Zahlen

Zahlen pro Tag:

- 2.100.000 Anwendungen (Fenster)
- 6.300.000 C/S-Kommunikationen
- bis zu 2.000.000 Messages
- 4.200.000 Objekt-Transaktionen
- 30.000.000 Großrechner-Transaktionen (DB-Transaktionen)
- 260.000.000 BO-Instanzen
- 4.800.000.000 Feldzugriffe auf BOs (geschützt und transaktional)

Sehr viele Zugriffe auf viele BOs mit wenigen DB-Transaktionen durch sehr wenige C/S-Kommunikationen.



Framework für Business Objects

Laufzeitumgebung für BOs

- Leichte BOs (POJOs) => feingranulare Modellierung möglich
- Objekt-Transaktion fast mehrere DB-Transaktionen zusammen
- Concurrency-Handling über Datenbank (optimistisches Locking) => Verantwortung für ACID bei der Datenbank
- Angelehnt an Java Data Objects (JDO), ähnlich EJB3

Persistenz

- Mapping für DB2 und IMS-DB
 - „alte“ Datenbanken in „neue“ Objekte
- Viele Optimierungsmöglichkeiten (z.B. Prefetching)
- Generische Referenz – einheitliche Objekt-ID (Typ „Object“ in DB)

Performance der Persistenz sehr kritisch!

Das unternehmensweite OO-Modell (1)

- Kein zentral entworfenen OO-Modell
 - Modell wächst mit den Projekten
 - Architekten koordinieren
- Fachliche Gruppierung von Klassen (Komponenten)
- Frage: Welche Klasse/Komponente ist „verantwortlich“?
- Konsequentes Abhängigkeitsmanagement (Komponenten)
 - WER darf WEN nutzen: definieren und prüfen
 - Unterstützt auch gutes Design
- Fachliche Modelle „konkret“, nicht „generisch“
- „Harte“ Typisierung
 - Compile-Time-Safety: verhindert viele Fehler
 - Abhängigkeiten sichtbar

Das unternehmensweite OO-Modell (2)

- In Schnittstellen (Interfaces) denken
 - Implementierung ist eine Black-Box, „darf“ Nutzer nichts angehen
- Viele Werttypen und Aufzählungstypen (unveränderbar)
 - Helfen BOs schlank zu halten
 - Einfache Wiederverwendung
 - Beispiele: Anrede, Geschlecht, Bankverbindung, BLZ, Geldbetrag, ...
- Generische Referenzen ermöglichen flexible Querschnittssysteme
- „Keine“ echte Vererbung: Interface-Erweiterung und Delegation besser
- Framework muss Freiheit zur Modellierung schaffen

Resümee: Realisierung von gutem fachlichen Design ist schwieriger als die technische Basis zu schaffen

Integration auf dem Client: Anwendungs-Portal

- Abstraktion von Anwendungen: TKeasy, SAP, Centura, Office, ...
- Qualifizierte Übergabe von Parametern

Vers. Nr.	Status	Name	PLZ	Ort	Straße	LGst	BDst	B
0410638557	MG	Ten, Jsu	21109	Hamburg	Jamestr. 22 KL	0506	0564	07
2707628390	KI	Ten, Kind	22085	Hamburg	Hofweg 22	0506		
0410630068	MG	Ten, Klaus	22085	Hamburg	Hofweg 22	0507	0507	
0410630106	MG	Ten, Klaus	22041	Hamburg	Bärenallee 25	0506	0564	
0410637135	MG	Ten, Klara	22085	Hamburg	Hofweg 22	0506	0564	
0410930045	MG	Ten, Maik	22041	Hamburg	Bärenallee 25	0506	0564	
0410630743	MG	Ten, Manfred	22085	Hamburg	Hofweg 22	0506	0564	
0410630123	MG	Ten, Manfred	22769	Hamburg	Stresemannst..	0506	0564	
2707621221		Ten, Martin	22085	Hamburg	Hofweg 22	0046	0046	
	MG	Ten, mit RM				0058	0001	
0410637144	MG	Ten, Jsu	22085	Hamburg	Hofweg 22	0046	0046	

11

Erfahrungen

- SUN Java (J2SE) auf Windows und Solaris
 - Stabil für hochlastige Enterprise Systeme
 - Konfiguration des Garbage-Collectors schwierig (GC-Durchsatz pro Server und Tag: ca. 5 Terra-Byte)
 - 64-Bit: Speicherverbrauch zu hoch
- SWING
 - Für große Anwendungen mit hohen Ansprüchen an die GUI geeignet
 - Viele Möglichkeiten: für effiziente Entwicklung beschränken
 - Performance gut, hoher Speicherverbrauch

12

Erfolgsfaktoren

- Kein Big-Bang: Lernen durch schrittweise Umstellung
- Frühe Produktivität
- Bereitschaft zu Re-Designs – Erfahrungen auch umsetzen
- Markttrends sehr kritisch würdigen
 - „Bringt es mir wirklich etwas?“
 - „Kann es die Versprechen halten?“
- Schlanke Architektur
- Stabilität durch Architektur sicherstellen (einige externe Bibliotheken nicht „stabil“)
- Flexibler und performanter Persistenzdienst
- Integration: Anwendungen und Systeme
- Automatisierte Qualitätssicherung (auf Source- und Byte-Code)

Vielen herzlichen Dank für
Ihre Aufmerksamkeit!

Ralf.Degner@tk-online.de

