

## Persistenz Services in der Service Component Architecture



### Die nächste Generation

### Anforderungen an zukünftige Persistenzmechanismen

Andreas Holubek  
Chief Architect

**Signsoft GmbH**  
Leipziger Str. 118  
01127 Dresden  
p +49 (0)351 894 53-0  
f +49 (0)351 894 53-29  
e a.holubek@signsoft.com  
w www.signsoft.de

0

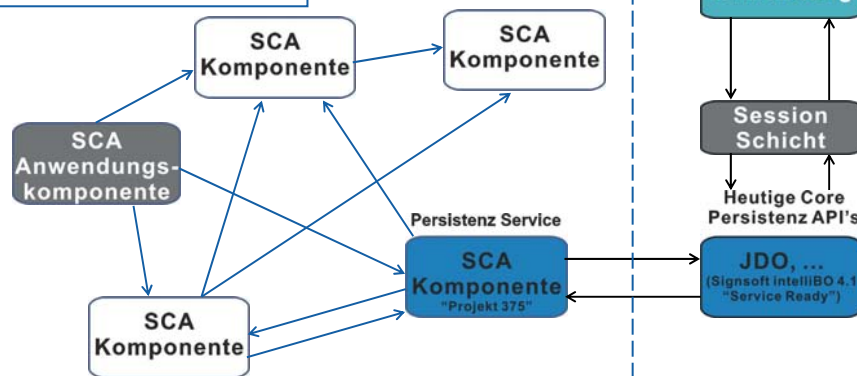
## Überblick

- Service Component Architecture
- SCA und Persistenz (Web Services)
- Vorhandenes und APIs
- Neue Ziele und Architekturen
- Neue Probleme
- Was ist heute machbar?
- Fazit und Ausblick
- Live Demo

1

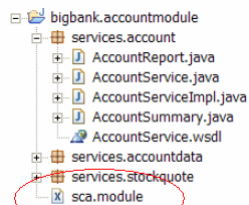
- Spezifikationen die Modell für die Anwendungsentwicklung beschreiben
- Basiert auf der Service Oriented Architecture (SOA)
- Sprachunabhängig (C++, Fortran, Java, PHP, ...)
- Keine Beteiligung an sprachabhängigen Prozessen (Java Community Process, ...)
- Separierung der Verantwortlichkeiten (Security, Transactions, ...)
  
- SDO für Parameter und Rückgabewerte (Service Data Objects)
- Java und C++ API Bindings sind vorhanden (v0.9)

**Anwendung:**  
- Auswählen der Services  
- Services miteinander verbinden



- Entwicklung von Standardkomponenten
- Vereinfachte Entwicklung und Verteilung von Anwendungen als Netzwerk von Services
- Erhöhte Flexibilität
- Erhöhte Testbarkeit
- Unabhängig von Sprache, System damit offen für die Zukunft
  
- Stand (!): November/Dezember 2005

- SCA Komponentenbeschreibung
- Verbindungspunkte zu anderen Komponenten



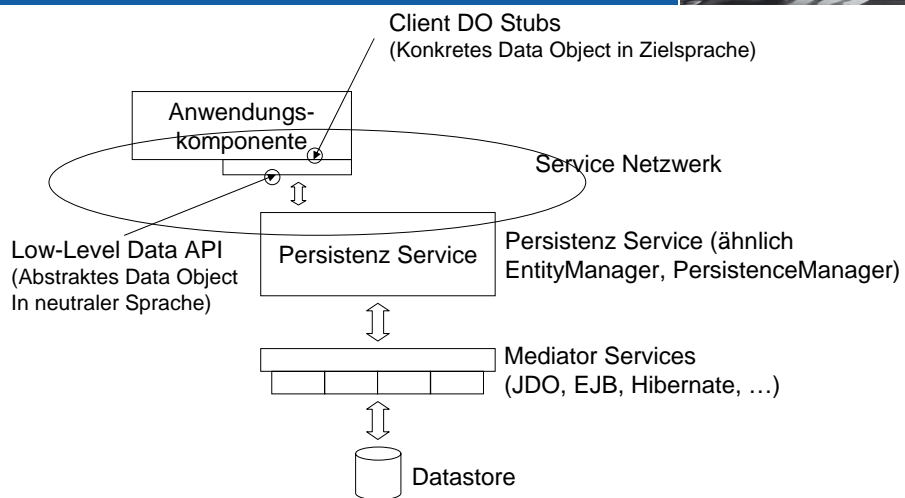
## Persistenz Service - Ziele



- Daten; überall zu jeder Zeit in jeder Sprache
- Transparenz auch im Client; Lazy Loading unabhängig vom Adressraum
- Transaktionssicherheit
- Anwendungen „die überleben“

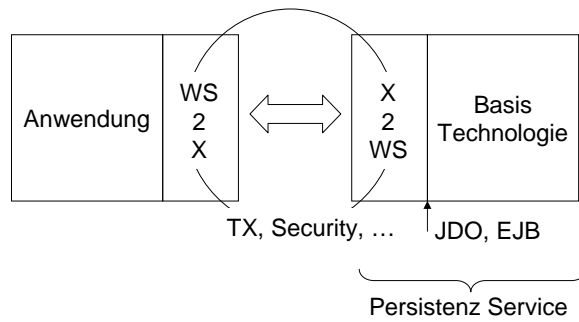
6

## Persistenz Service – Architektur (Signsoft)



7

- Einführung von Schichten zur Transformation
- Etablierung von begleitenden Diensten
- Zukunft: Reduzierung der Schichten



8

Vorhandenes und warum es nicht ausreicht.

9

- **Anforderung: Persistenzprovider sollte Web Services direkt unterstützen.**  
Derzeitig keine 100%ige Lösung in Sicht.

- **Betrachtung der vorhandenen Basis**

- EJB 3 – Persistenz
- JDO 2
- SDO

- **Work in Progress**
- **Standard Attach/Detach Mechanismus**
- **Kein Austauschformat**
- **Kein Lazy Loading, Attributerkennung auf Client Seite**
- **JSR (noch in Arbeit)**

## Java Data Objects (JDO)



- Work in Progress
- Standard Attach/Detach Mechanismus
- Kein Austauschformat
- Kein Lazy Loading, Attributerkennung auf Client Seite
- JSR (Zukunft offen)

12

## Service Data Objects (SDO)



- Austauschformat ist definiert (XML)
- Hybride Datenmodelle möglich
- Kein Lazy Loading, Attributerkennung auf Client Seite
- Nur Teilgraphen ohne Bidirektionalität
- Implementierungen vorhanden
- Standard (2.x) wird von Firmengruppe definiert
  
- Problem und Vorteil: Sprachunabhängige Datenobjekte

13

- **Transparenz**
- **Transaktionen**
- **Sicherheit**
- **Performanz**

- **Ad-hoc-Lösungen wie z.B. die Veröffentlichung von Funktionen eines Persistenzmanagers als Web Service können keine umfassenden Lösungen im Sinne dieser Problemkreise bieten.**

**Transparenz - auch im Client  
(Anforderung Part 1)**

## Problem: Transparenz im Client



### ■ Value Objects, Data Transfer Objects

- Gängige (J2EE) Pattern
- „Alter Hut“; wird benötigt, aber:

- Lazy Loading?
- Teilgraphen?
- Dynamik?
- Lebensdauer?
- Austauschbarkeit?

### ■ Übertragen werden lediglich unvollständige Objekte

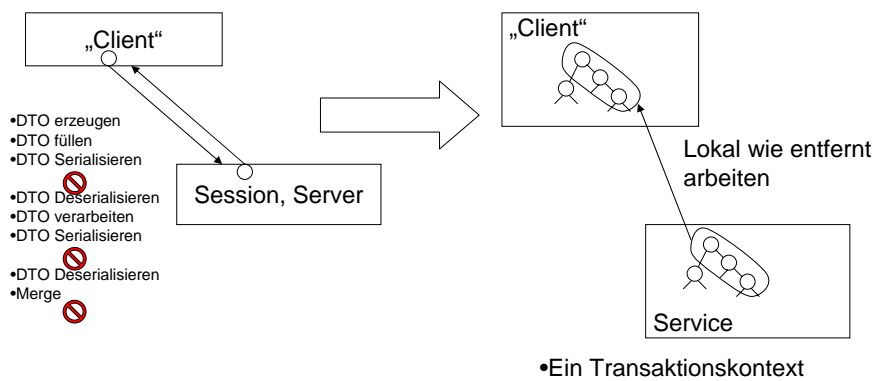
### ■ Versuchte Lösung: Attach, Detach, Merge, ...

16

## Ansprüche für die Zukunft



### ■ Transparenz im Client, in der Session



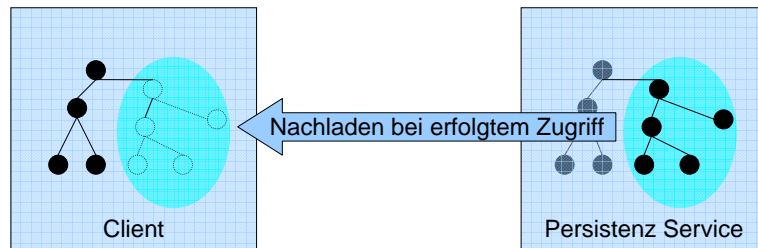
17

### Datenobjekte, DTO sind unabhängig von System und Sprache.

- Aber, werden heute sehr eng daran gekoppelt.
  - Nutzbarkeit, Domainmodell ist an die Zeit, Firma, Strategie,... gebunden (!).
  - Hybride Datenmodelle gewinnen an Bedeutung.
- **Lösung: Beschreibung und Transport in einer neutralen Form => XML.**

### ■ Mit Metainformationen angereicherte XML- Repräsentation der Objekte

- > Verarbeitung z.B. durch JAX-RPC Handler Chains
- > Clientseitiger Aufbau ganzer Objektgraphen
- > Zugriffsabhängiges Nachladen von Attributen und Teilgraphen



**Transaktionen  
(Anforderung Part 2)**

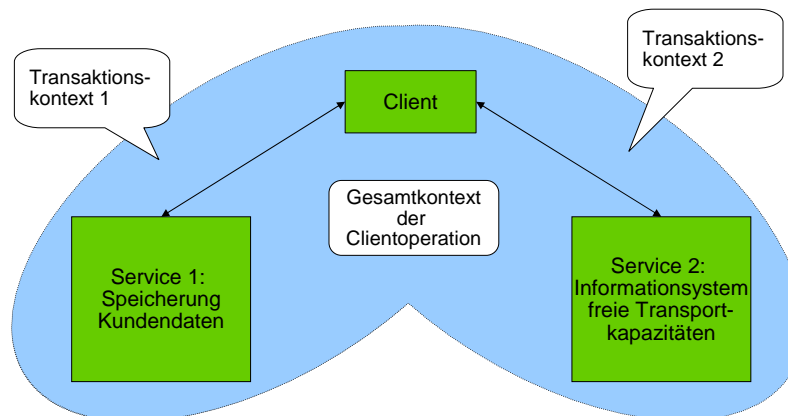
■ **Transaktionskontext zusammengesetzt aus:**

Transaktion(en) des gemeinsamen Persistenzkontextes  
von Client und Persistenzdienst

+

Transaktionskontext über alle beteiligten Services  
der durch den Client initiierten Operation

■ **Beispiel:**



- „Web Services Coordination” als mögliche Basis für notwendigen Web Services-Transaktionskontext
- August 2005 – Version 1.0 vorgelegt von Arjuna, BEA, Hitachi, IBM, IONA, Microsoft

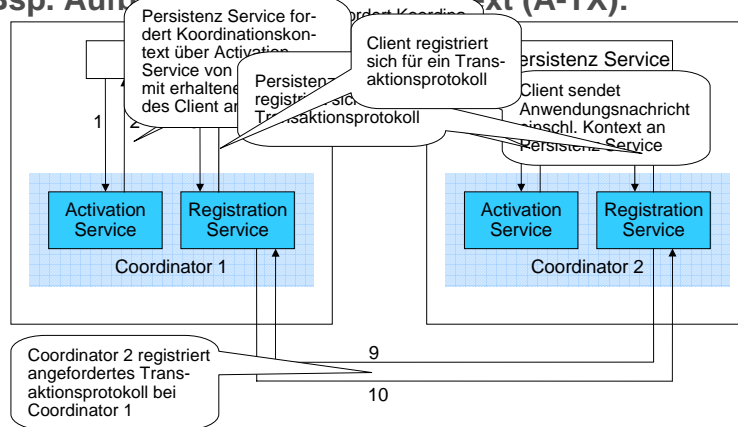
- **Konzept:**
  - Bereitstellen einer (erweiterbaren) Basisplattform zur Koordination von Web Service Interoperabilität
  - Klassifikation von Koordinationen durch verschiedene Koordinationstypen
  - Koordinationstypen-abhängiger Umfang verfügbarer Transaktionsprotokolle
  - Management verteilter WS-Transaktionen mithilfe von Coordinator-Instanzen

### ■ Koordinationstypen:

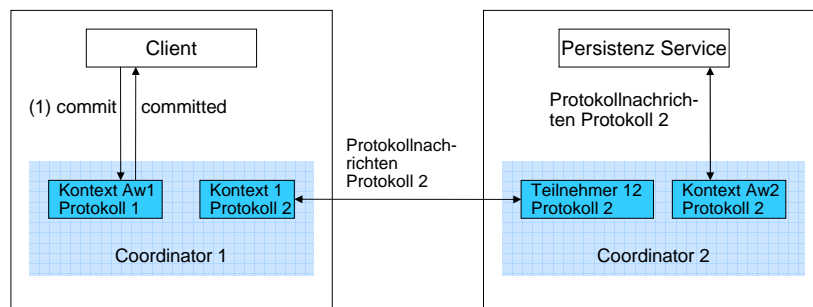
- Web Services Atomic Transaction
  - Kurzlebige Verbindungen
  - Einhaltung von ACID-Kriterien
  - Vertrauenskontext zwischen den Teilnehmern notwendig
  - Transaktionsprotokolle: Completion, 2PC

- Web Services Business Activity Framework
  - Langlebige Verbindungen
  - Vertrauenskontext zwischen Teilnehmern kann nicht vorausgesetzt werden
  - u.U. Zusammengesetzt aus einzelnen Atomic-TX
  - Zwei Koordinationsprotokolle (BusinessAgreementWithParticipantCompletion, BusinessAgreementWithCoordinatorCompletion)
    - Erfolgreicher Transaktionsabschluss nicht in jedem Fall vom Erfolg der Operationen aller Teilnehmer abhängig (Aufweichung ACID, Kompensation)

■ Bsp. Aufbau Koordinationskontext (A-TX):



■ Bsp. Abschluss Transaktion (A-TX)



## Sicherheit und Performanz (Anforderung Part 3 und 4)

30

## Sicherheit

- **Schutz der Daten in verteilter Systemumgebung vor:**
  - Einsichtnahme Unbefugter
  - Änderung durch Unbefugte
- **Zugriff auf Persistenzdienst nur für autorisierte Instanzen**
- **Unabstreitbarkeit gesendeter Nachrichten**

31

### ■ Mögliche Lösungsansätze:

- HTTP als Quasistandard bei Wahl des Transportprotokolls für SOAP-Nachrichten
  - Einsatz sicherer Kanäle (HTTPS) zum Schutz versendeter Daten
- Authentifizierung und Autorisierung mithilfe von WS-Security

- Performanzeinbruch durch XML-Repräsentation der zu übertragenden Daten?
- Steigende Leistungsfähigkeit moderner IT-Systeme
- Zahlreiche Möglichkeiten der Optimierung
  - Parsing von SOAP-Nachrichten
  - Reduktion von Netzwerkzugriffen
  - Übertragung von (Teil-)Graphen persistenter Objekte

## Realisierung

34

## Idee Persistenz Service

### ■ Umsetzung von hierarchisch strukturierten Services zur Persistierung von Objekten

- Veröffentlichung von Service zur Objektpersistierung (initiale Aufrufchnittstelle für den Client)
- Persistenz Service ist Menge persistenter Objekte mit „Servicecharakter“ untergeordnet

35

- **Apache Axis/Axis2**
- **Web Services Coordination**
  - Web Services Atomic Transaction
  - Web Services Business Activity Framework
- **JAX-RPC/Axiom**
- **WS-Security**
- **Persistenztechnologien wie JDO/EJB3-Persistenz**

- **Umfeld für Lösungen existiert – Umsetzung erfordert jedoch noch viel Handarbeit**
- **Perspektivisch: Abbau der Service-seitigen Schicht und Ersatz der darunterliegenden Basistechnologie durch XML-Repräsentation**
- **Clientseitige Objektorreicherung – weg von „dummen“ Datenobjekten, hin zu Funktionalität**
  
- **SCA als zukünftiges Architekturmodell**
- **Keine Abhängigkeit von JCP, J2EE, .NET, ...**

## Web Services Transaktionen

<http://www-128.ibm.com/developerworks/library/specification/ws-tx/>

- Web Services Coordination  
<ftp://www6.software.ibm.com/software/developer/library/WS-Coordination.pdf>
- Web Services Atomic Transaction  
<ftp://www6.software.ibm.com/software/developer/library/WS-AtomicTransaction.pdf>
- Web Services Business Activity Framework  
<ftp://www6.software.ibm.com/software/developer/library/WS-BusinessActivity.pdf>
- Business Transaction Protocol  
[http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=business-transaction](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=business-transaction)

## Apache Axis

- <http://ws.apache.org/axis/>
- <http://ws.apache.org/axis2/>
- Dapeng Wang (Hrsg.), Thomas Bayer, Thilo Frotscher, Marc Teufel. Java Web Services mit Apache Axis. Software & Support Verlag, 2004.

## Web Services Security

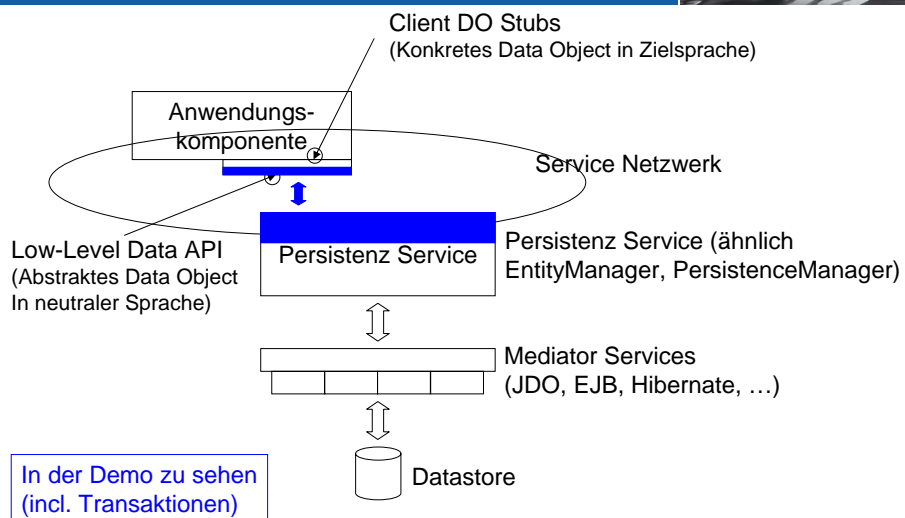
- <http://www.oasis-open.org/specs/index.php#wssv1.0>

## JAX-RPC

- <http://java.sun.com/webservices/jaxrpc/index.jsp>

## SCA

- <http://www-128.ibm.com/developerworks/library/specification/ws-sca/>



**Vielen Dank!**

**Fragen?**