

ICT Strategie + Führung

Service Oriented Architecture

Übung 16.3: Was ist SOA

- Standardisierung der IT zur Unterstützung von Geschäftsprozessen
- Orientierung der IT an Geschäftsprozesse, deren Abstraktionsebenen die Grundlage für konkrete Serviceimplementierungen sind.
- Beispiel: „Vergib einen Kredit“ ist beispielsweise auf einer hohen Ebene angesiedelt, dahinter verbirgt sich bei einem Bankunternehmen ein Geschäftsprozess mit einigen beteiligten Personen und informationstechnischen Systemen („Eröffnen der Geschäftsbeziehung“, „Eröffnen eines oder mehrerer Konten“, „Kreditvertrag...“ und so weiter), während „Trage den Kunden ins Kundenverzeichnis ein“ ein Dienst auf einer niedrigeren Ebene ist.
- Durch Zusammensetzen (Orchestrierung) von Services niedriger Abstraktionsebene können so recht flexibel und unter Ermöglichung grösstmöglicher Wiederverwendbarkeit Services höherer Abstraktionsebenen geschaffen werden.
- Die Kommunikation zwischen Nutzer und dienstleistender Komponente wird über einen Enterprise Bus oder über Peer to Peer Protokolle realisiert.

Quelle: http://de.wikipedia.org/wiki/Serviceorientierte_Architektur

SOA Fragen

1. Welches sind die wichtigsten Treiber für SOA (Service Oriented Architecture)?
 - Verteilte Systeme, schlechte Integration
 - Datenredundanzen und damit verbundene Mehrkosten
 - Schneller Wandel der Anforderungen seitens Business (Time to Market)
 - Langsame Change-Umsetzung wegen grossen Abhängigkeiten und grosser Komplexität
 - Hohe Kosten für IT-Entwicklungen (hohe Ausgaben für Schnittstellen-Anpassungen)
2. Warum besteht heute in grösseren Umgebungen ein grosser Integrationsbedarf?
 - Schnellere Reaktion auf Änderung der Geschäftsanforderung
 - Bedarf höherer Automatisierung um Effizienz und Effektivität zu steigern
 - Intern verwendete Prozesse können über definierte Schnittstellen auch extern genutzt werden (B2B und B2C)
3. Wo sehen Sie die grössten Probleme in einer stark heterogenen, verteilten aber dennoch „verknüpften“ Systemlandschaft?
 - Komplexe und „statische Schnittstellen“ verursachen hohen Aufwand bei Änderungen.
Beispiel: Verschiedene Anwendungen greifen auf eine Datenbank zu. Eine Erweiterung/Anpassung des Datenbankschemas würde auch eine Anpassung der Anwendungen bedingen.
(Stichwort: Starke Kopplung)
 - Gleiche Funktionalitäten werden an verschiedenen Orten implementiert. Änderungen müssen an mehreren Stellen nachgefahren werden.
(Stichwort: Niedrige Kohäsion)
 - Eine Vielzahl von Schnittstellen ist schwieriger zu handhaben als eine einheitliche, für alle geltende Schnittstelle.
 - Standards bei der Schnittstellenentwicklung kann dem Verständnis von neuen Schnittstellen helfen.
4. Welches wären mögliche Lösungsansätze?
 - Datenintegration: Datenbanken mit Schnittstellen und Vermittlungsdiensten entkoppeln. Message Queueing Systeme, Plattform-unabhängig.
 - Methodenintegration: Applikationen rufen Methoden über Middleware Systeme auf. Die Middleware Systeme implementieren die Geschäftsprozesse oder nutzen Systeme, welche Geschäftsprozesse implementieren.
 - Prozessintegration: Kleine modulare und flexibel zusammensetzbare Services, welche die Geschäftsprozesse abbilden sollen.
 - Integration über User Interface: Verstecken mehrerer Anwendungen hinter einer einheitlichen Benutzerschnittstelle.

5. Was sind die Lösungsansätze für die Integration von grossen und komplexen Systemumgebungen von R. Zacharias und H. Glöckle.

→

Übung 3: Service Design

Die Payment-Komponente soll für die Lohnapplikation einen „Service“ offerieren, welcher es erlaubt eine Lohnzahlung zu kreieren. Typische Parameter sind:

- Lohnkonto des Mitarbeiters (Kredit-Account)
- Betrag
- Ausführungsdatum

Wie würden Sie das tun?

→ Schnittstelle definieren. In dieser Schnittstelle könnte beispielsweise folgende Methode definiert werden:

```
Lohnzahlung(      IN LohnkontoNr: account,
                  IN Betrag: double,
                  IN Ausführungsdatum: date
                  OUT TransaktionsID: long      )
```

Ist das gutes Schnittstellen-Design? Wo könnten Probleme auftreten?

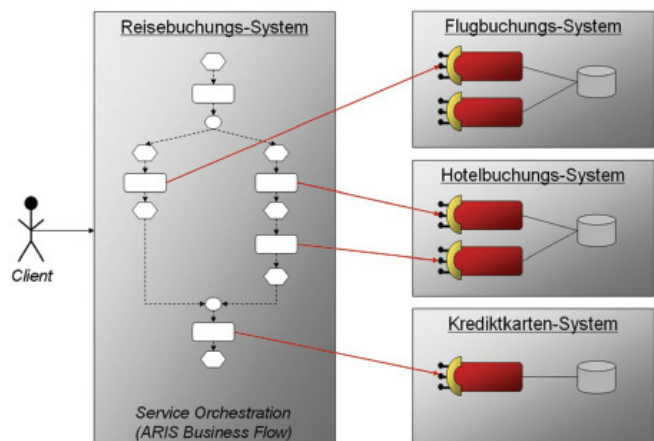
→ Die Schnittstelle ist zu schmal und zu allgemein (nur für Lohnzahlungen). Eine Erweiterung und Generalisierung könnte die Schnittstelle universeller machen.

```
Zahlung(         IN Quellkonto: account,
                  IN Zielkonto: account,
                  IN Betrag: double,
                  IN Währung: currency,
                  IN Ausführungsdatum: date
                  OUT TransaktionsID: long      )
```

Gute Wiederverwendbarkeit muss erreicht werden. Es zeugt von schlechtem Applikationsdesign, wenn mehrere ähnliche/unabhängige Schnittstellen existieren.

Übung 4.1: SOA Konzept

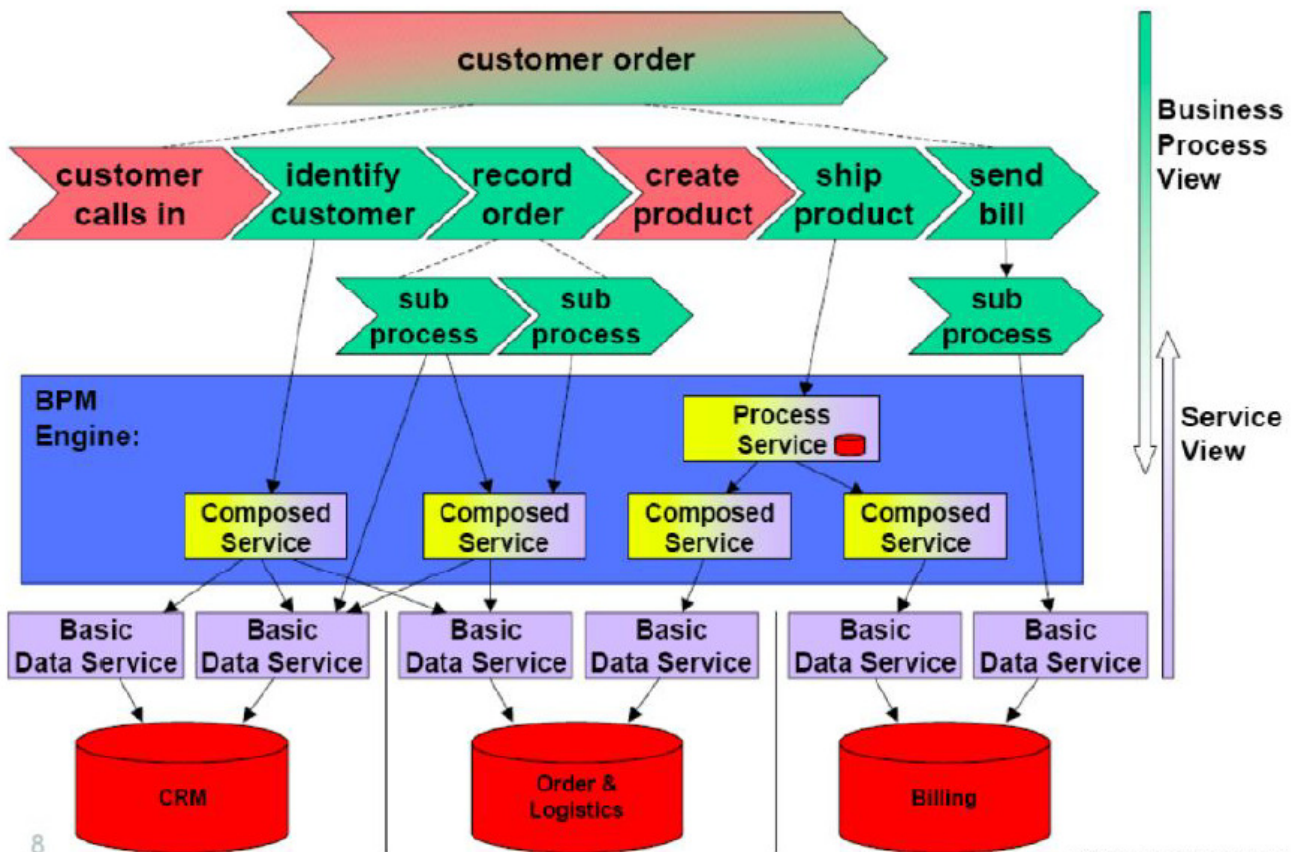
- Schnittstelle stellt Vertrag zwischen Dienstanbieter und –nutzer dar.
- Operationen sind einzelne logische Arbeitseinheiten, die letztendlich meist direkt auf technische Transaktionen abgebildet werden.
- SOA stellt eine verteilte Systemarchitektur mit föderativen Diensten dar. Dienste sind im Netz auffindbar und implementieren oft einen bestimmten Geschäftsprozess.
- Der Kontext, in welchem ein Dienst genutzt wird, muss nicht bekannt sein.
- Dieselben Funktionalitäten können immer wieder verwendet werden. Das wirkt sich positiv auf die Wartbarkeit, Erweiterbarkeit und Robustheit aus. Indirekt natürlich auch auf die Kosten.
- Im Idealfall ist SOA plattform-, sprach- und betriebssystemunabhängig.
- Heutige Architekturen sind oft prozessorientiert. Das Prozesswissen ist im ganzen System verstreut. Der OO-Ansatz „alles ist ein Objekt“ verstärkt diese Problematik noch mehr.



Quelle: Roger Zacharias, „Der OO-König ist tot, es lebe der SOA-König“

Was ist Service Orchestrierung?

Mehrere SOA Dienste werden zu einem übergeordneten Geschäftsprozess zusammengestellt.



8

Übung 4.2: Einschätzung und Bewertung von SOA

Lesen Sie den Text „Einschätzung und Bewertung von SOA (SWOT Analyse)“ sorgfältig durch. Prüfen Sie, welches die wichtigsten Aussagen sind und fassen Sie die (potenziellen) Positiv- und Negativpunkte zusammen, indem Sie sie möglichst einander gegenüberstellen (in Form einer Tabelle oder Punkteliste).

	Vorteil	Nachteil
Skalierbarkeit und Investitionsschutz	<ul style="list-style-type: none"> • Gute Skalierbarkeit dank starker Entkopplung und Message Queuing • Nutzbarkeit bestehender Entwicklungen verlängert 	<ul style="list-style-type: none"> • Niedrigere Performance wegen starker Entkopplung und Transformationen von Messages • Overhead von Messages
Akzeptanz	<ul style="list-style-type: none"> • Problematik allg. erkannt • Viele (namhafte) Hersteller beteiligt 	<ul style="list-style-type: none"> • Begriffs-Chaos: SOA wird mit Web Services gleichgestellt
Umsetzung neuer Geschäftsprozesse	<ul style="list-style-type: none"> • Neue Kombinationen bestehender Services können neue Prozesse unterstützen • Time-to-Market sinkt 	<ul style="list-style-type: none"> • Bei der Umsetzung müssen bereits zukünftige Anforderungen berücksichtigt werden.
Komplexität	<ul style="list-style-type: none"> • Entkopplung einzelner Komponenten senken Komplexität innerhalb jeder Komponente • Implementation wird hinter Schnittstelle verborgen 	<ul style="list-style-type: none"> • Verlust von Synergien durch Parallelentwicklungen (Fordert Methoden- und Architekturmanagement) • Erfordert gute Dokumentation der Schnittstellen • Erfordert fein-granulare SLA's für jeden angebotenen Service

Übung 4: SOA Prinzipien

Welche Ihnen bekannten Architekturprinzipien spielen bei SOA eine wichtige Rolle?

- Das Konzept von „Information Hiding“.
- Modularisierung, Bildung von Komponenten
- Trennung der Verantwortlichkeiten
- Lose Kopplung (Austauschbarkeit, Wartbarkeit) und hohe Kohäsion (enger innerer Zusammenhalt)
- Client-/Server Prinzip: Anwendungen werden zentral betrieben. Kommunikation über Request/Response Verfahren.