TECHNISCHE UNIVERSITÄT DRESDEN

Vorlesung
“Service and Cloud Computing"

6. SaaS – WS-Erweiterungen und SOA-Infrastruktur

Dr.-Ing. Iris Braun

TECHNISCHE UNIVERSITÄT DRESDEN

Vorlesung "Service and Cloud Computing"

6. SaaS - WS-Erweiterungen und SOA-Infrastruktur

6.1 WS-*-Erweiterungen

Dr.-Ing. Iris Braun

Gliederung

- WS-* Erweiterungen
  - Adressierung
  - Zuverlässige Nachrichtenübertragung
  - Asynchrone Nachrichtenübertragung
  - Transaktionen
  - Festlegung von Richtlinien zur Dienstnutzung
- Frameworks zur technischen Umsetzung einer SOA
  - ESB – Enterprise Service Bus
  - JBI – Java Business Integration
  - SCA – Service Component Architecture

Motivation: Rückblick

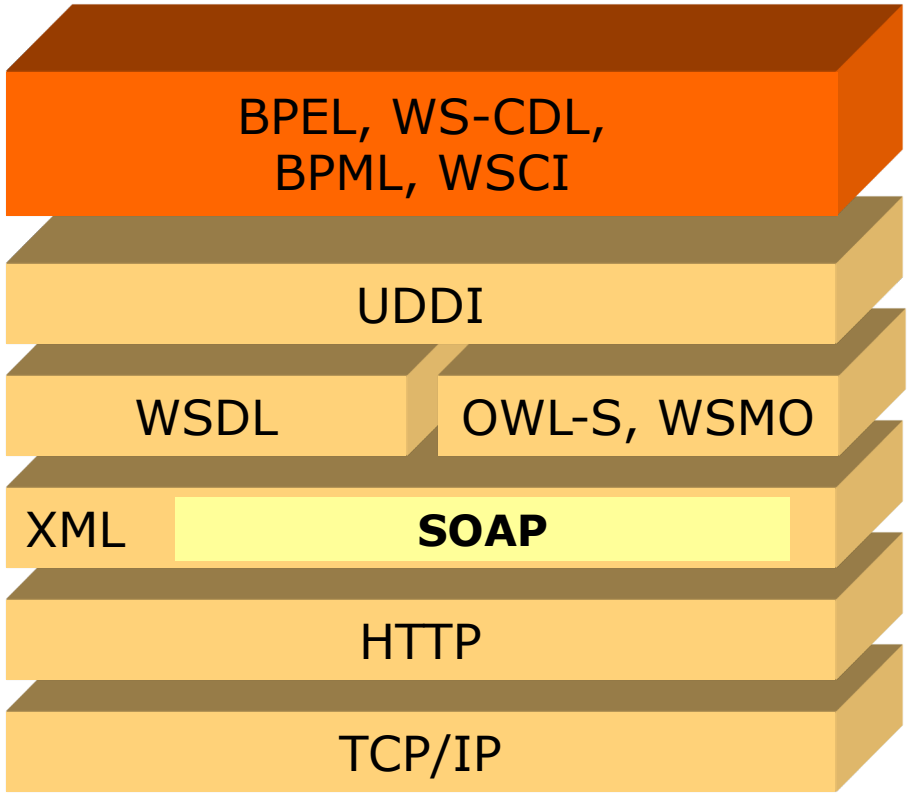
Geschäftsprozess

Verzeichnis

Dienstbeschreibung

Nachrichtenformat

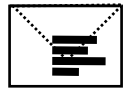
Nachrichtenprotokoll



Problem: Grundlegende WS-Spezifikationen nicht ausreichend, um alle Anforderungen der Geschäftsprozess-Schicht zu erfüllen!

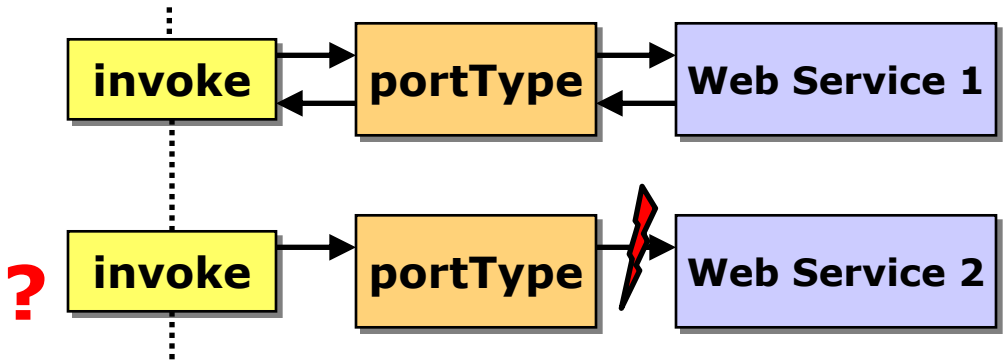
Motivation: Beispiele für offene Fragen

Wie kann man in einer SOAP-Nachricht angeben, von wem sie kommt, wer sie empfangen soll, und an wen die Antwort geschickt werden soll?



Wie wird sichergestellt, dass eine Nachricht tatsächlich ankommt und nur einmal verarbeitet wird?

Was passiert in einem Geschäftsprozess mit einzelnen Aktivitäten, die erfolgreich ausgeführt werden, während andere Aktivitäten fehlschlagen?



Architekturansatz

Geschäftsprozess

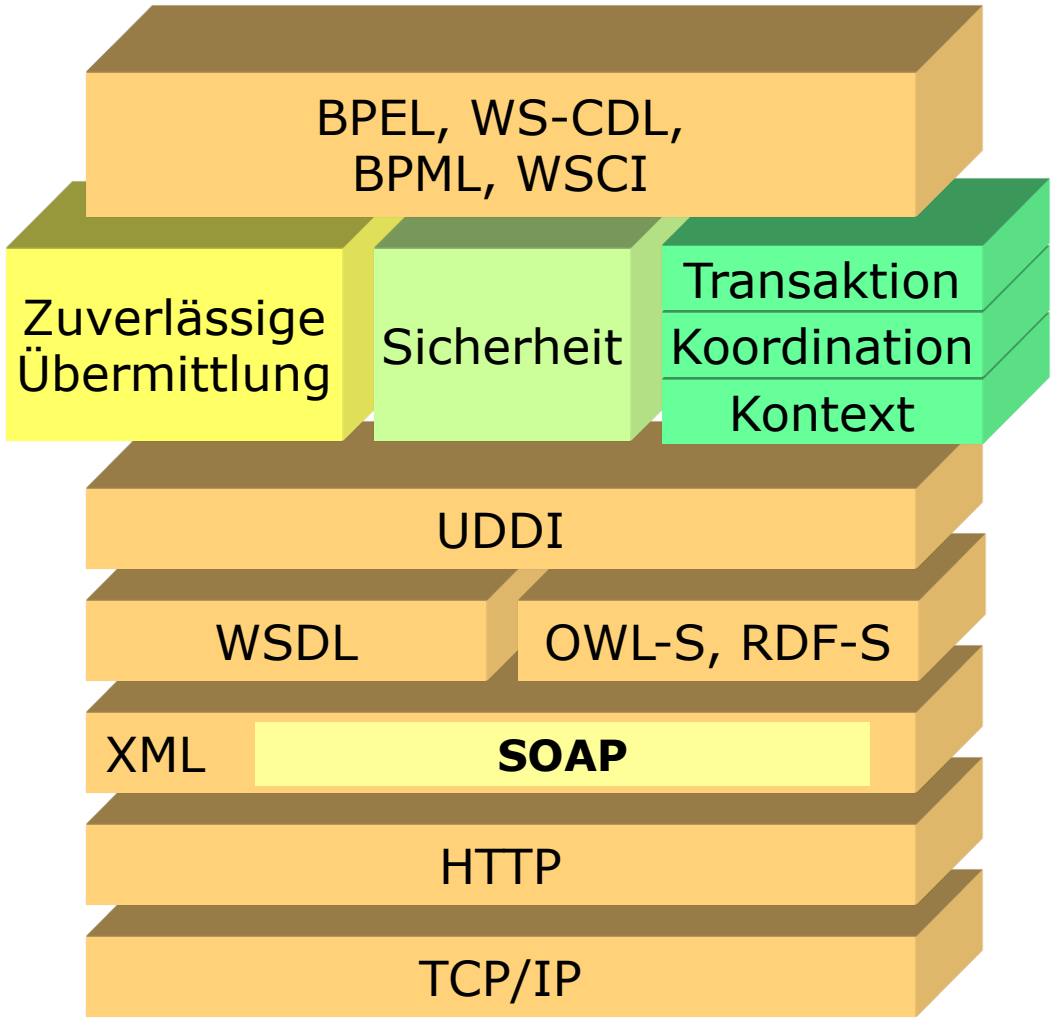
„WS-* Erweiterungen“

Verzeichnis

Dienstbeschreibung

Nachrichtenformat

Nachrichtenprotokoll



Ziele

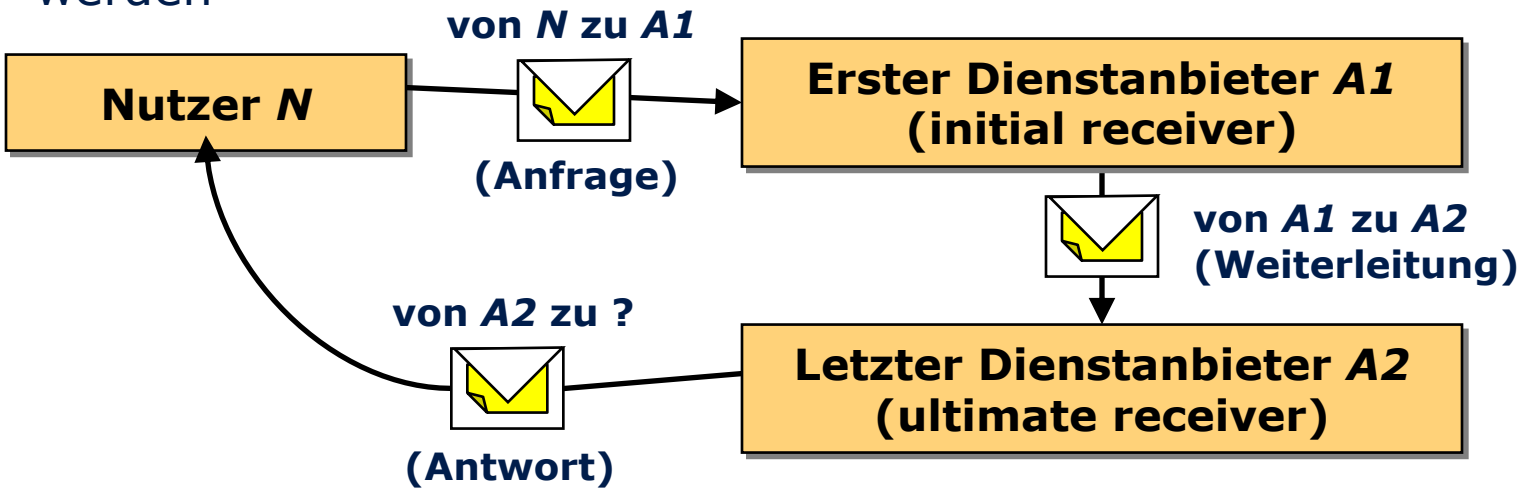
„WS-* Erweiterungen“

- Grundlagen für komplexe Geschäftsprozesse definieren
- Standardisierung, Unabhängigkeit von Transportprotokollen

- *Heute:* Adressierung, zuverlässige Nachrichtenübermittlung, Benachrichtigung, Transaktionen

Adressierung

- Szenario: Mehrere Web-Service-Anbieter arbeiten zusammen
  - Anfragekette zwischen Dienstanbietern, Antwort soll von letztem Anbieter *A2* in dieser Kette zum Nutzer *N* übermittelt werden



  -> *A2* muss dafür die Adresse des ursprünglichen Dienstnutzers *N* kennen

- Adressierung auch hilfreich für zuverlässige Nachrichten-übertragung bei Problemen(Netzwerkfehler, Antwort-Verlust, ...)

WS-Addressing

- W3C-Standard, Version 1.0 (2006)
- Ersetzt frühere Spezifikationen WS-Referral und WS-Routing
- Transport-neutrale Methode zum Ansprechen von Objekten (Endpunkt-Referenz)
- Erweiterung für SOAP-Header, definiert
  - Endpunkt-Beschreibung (Zugriffspunkt eines Dienstes im Netzwerk, zusätzliche zur Interaktion nötige Metadaten)
  - Message Information Headers (Elemente zur Adressierung)
- Abstraktion von Transportprotokoll, erleichtert Implementierung für mehrere Transportprotokolle („multi-transport“)
- WS-Addressing Action ersetzt das Action-Attribut des SOAP-HTTP-Bindings (in SOAP 1.2 entfernt)
  - Ermöglicht Aufruf eines generischen Dienstes („Bank“) mit einer bestimmten Operation (Action, z.B. „Überweisen“)

WS-Addressing: Beispiel

```
<soap:Header>
   <wsa:ReplyTo>
     <wsa:Address>
       http://www.user.de/testuser/TestSearchRequest
     </wsa:Address>
   </wsa:ReplyTo>
   <wsa:To>
     http://api.holtin.com/hss
   </wsa:To>
   <wsa:Action>
     http://api.holtin.com/hss/search
   </wsa:Action>
</soap:Header>
```

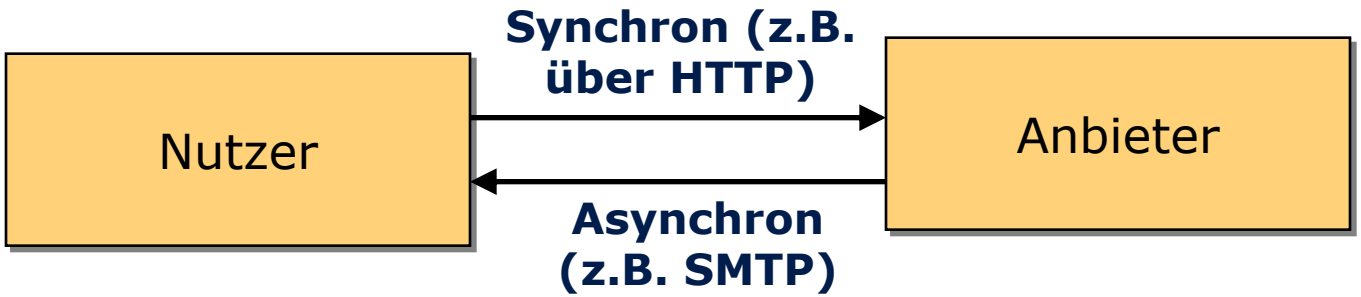
Antwort-Adresse

Adresse des Dienstanbieters

Auszuführende Operation

WS-MessageDelivery

- Spezifikation 2004 von BEA Systems, Microsoft, IBM und Tibco an W3C übergeben, Version 1.2 (2009)
- Definiert
  - Nachrichtenidentifikation
  - Abstrakte Übertragungseigenschaften
  - Nachrichtenreferenzierung für MEP (Nachrichtenaustausch-muster, z.B. Callback, Broadcast)
- Ziel: Abstraktion der SOAP-Nachricht von Transportprotokollen
- Ermöglicht Referenz auf Nachrichten, Verwendung von verschiedenen Protokollen innerhalb eines MEP



WS-MessageDelivery: Beispiel

```
... xmlns:wsmd="http://www.w3.org/2004/04/ws-messagedelivery"
<soap:Header>
   <wsmd:MessageOriginator ... >
     <wsdl:service ... > <wsdl:port ... >
        <soapbind:address location="http://..."/>
     </wsdl:port> </wsdl:service>
   </wsmd:MessageOriginator>
   <wsmd:MessageDestination>
     <wsmd:uri>http://example.com/Client-A/reply</wsmd:uri>
   </wsmd:MessageDestination>
   <wsmd:MessageID>uuid:5a389ad2-22dd-11d1-aa77-002035b29092
   </wsmd:MessageID>
   <wsmd:MessageReference wsmd:reason="&wsmd;/reason/response">
     uuid:58f202ac-22cf-11d1-b12d-002035b29092</wsmd:Message...>
   <wsmd:OperationName>myRequestResponseOperation</wsmd:Op...>
   ...
```

Vergleich WS-Addressing vs. WS-MessageDelivery

WS-Addressing	WS-MessageDelivery
From (EndPointRef)	MessageOriginator (wsdlLocation|service|uri|serviceQname)
To (URI)	MessageDestination (wsdlLocation|service|uri|serviceQname)
ReplyTo (EPR)	ReplyDestination (wsdlLocation|service|uri|serviceQname)
FaultDestination (EPR)	FaultTo (wsdlLocation|service|uri|serviceQname)
MessageID (URI)	MessageId (URI)
RelatesTo (URI)	MessageReference (URI)
Action (URI)	OperationName (NCName)

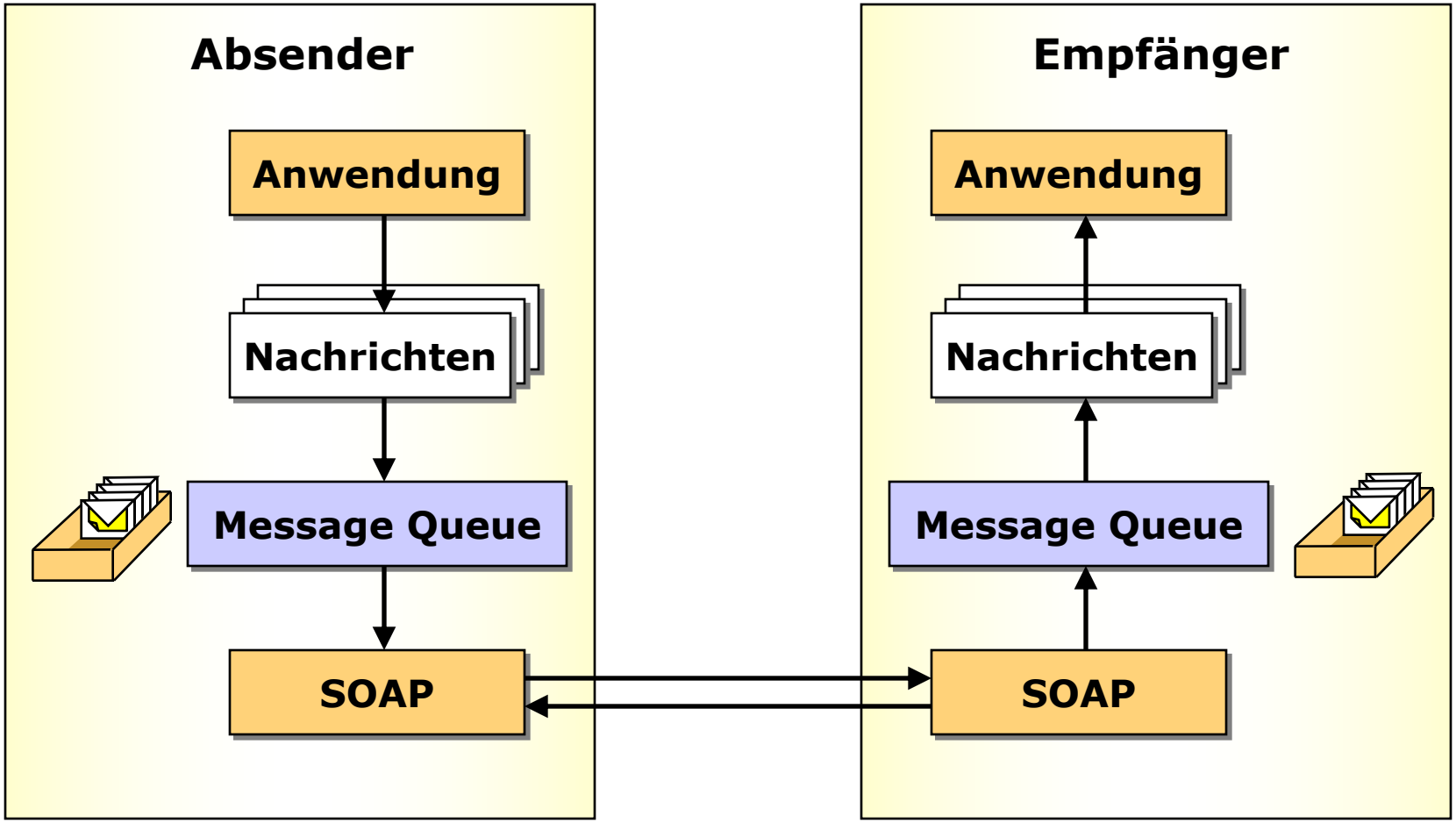
⇨ WS-Addressing gebräuchlicher, Basis weiterer Standards (WS-ReliableMessaging, WS-Policy, WS-Security,…)

Erweiterte Nachrichtenübermittlung

- Zuverlässige Übermittlung
  - Sehr wichtig für unternehmenskritische Anwendungen
  - Middleware-Transportprotokolle: zuverlässig (WebSphere MessageQueue (MQ), Java Message Service (JMS), Apache ActiveMQ, MQTT)

- Erweiterte MEPs
  - Ereignisbenachrichtigung
  - Publish-/Subscribe-Mechanismen
  - Techniken für mobile Anwendungen

Zuverlässige Übermittlung: Architektur



Zuverlässige Übermittlung: Konzepte

Grundlegende Reliable-Messaging-Konzepte:

- Garantierte Nachrichtenübertragung
- Benachrichtigung über den Nachrichtenstatus
- Eliminierung von Duplikaten (doppelt empfangene Nachrichten)
- Sortierung der Nachrichten (Reihenfolge Sender = Empfänger)

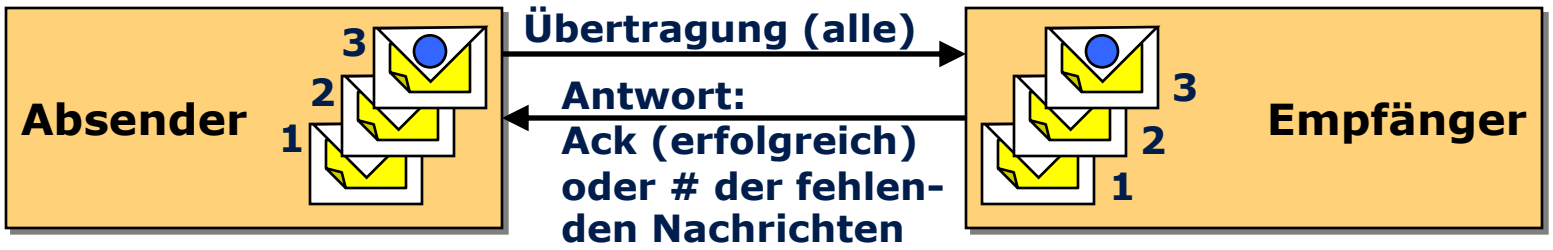
Kardinalitäts-Garantien:

- Mindestens einmal (at least once)
- Genau einmal (exactly once)
- Höchstens einmal (at most once)

„Normales" SOAP über HTTP: erfüllt keines dieser Konzepte!

Zuverlässige Übermittlung: Grundidee

- Software-Infrastruktur, installiert sowohl bei Nachrichten-Sender als auch bei Nachrichten-Empfänger
- Behandelt Fehler in der Nachrichtenübertragung auf einem möglicherweise unzuverlässigen Netzwerk
- Üblich: symmetrische (Sender- = Empfänger-)Implementierung
- Grundidee:
  - Sequenz-ID für Gruppe zusammenhängender Nachrichten
  - Nachrichten-ID für einzelne Nachrichten dieser Gruppe
  - Absender markiert letzte Nachricht
  - Empfänger bestätigt



Zuverlässige Übermittlung: Vorteile

- Losere Kopplung
- Trennung der Verantwortlichkeiten („separation of concerns“) führt zu einfacherem Anwendungscode
  - Reliable Messaging in der Infrastruktur, Anwendungscode enthält nur noch Geschäftslogik
- Einfachere Administration
  - Wartung von integrierten Anwendungen: während ein Teil A offline ist, läuft Teil B einer anderen Organisation weiterhin
- Ermöglicht asynchrone Kommunikation
- Sichert Unabhängigkeit vom Transportprotokoll
- Reduziert Komplexität bei Nutzung unzuverlässiger Netzwerke

WS-ReliableMessaging vs. WS-Reliability

- **WS-Reliability**: Spezifikation von Oracle, Sun u.a., 2004 bei OASIS eingereicht, aktuelle Version 1.1
- **WS-ReliableMessaging**: Entwicklung von Microsoft, IBM, BEA u.a., wurde 2005 ebenfalls zur Standardisierung an OASIS übergeben, aktuelle Version 1.2

- Sehr ähnliche Standards, aber nicht zueinander kompatibel

Unterstützte Anforderungen (WS-R und WS-RM):

- Mindestens einmalige Übertragung
- Höchstens einmalige Übertragung
- Eliminierung von Duplikaten ⇒ genau einmalige Übertragung
- Garantierte Nachrichtenreihenfolge

Vergleich WS-ReliableMessaging/WS-Reliability

Feature	WS-ReliableMessaging	WS-Reliability
Garantierte Übertragung, Duplikateliminierung, Sortierung, SOAP-Binding	√	√
Nachrichten-Status	√	×
HTTP-Binding	√	×
SMTP-Binding	×	×
Fehlerbehandlung	√	√ (SOAP-Faults)
Sicherheit	√ über WS-Security und andere Spezifikationen	√ über WS-Security
Anforderungen (Policies)	√ WS-RM Policy Assertion, über WS-Policy	√ über WS-Policy

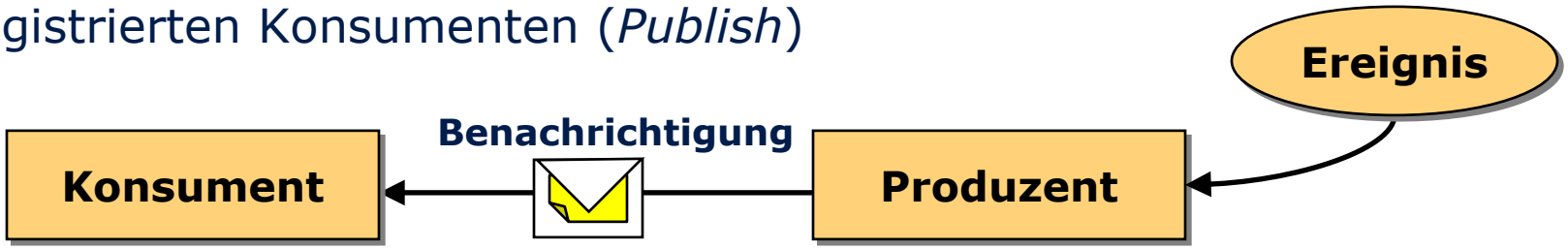
⇨ WS-RM hat breitere Industrieunterstützung (IBM, SAP, Microsoft, …)

Benachrichtigung (*Notification*)

Pub-Sub-Pattern

- Nachrichtenkonsument registriert sich (*Subscribe*)

- Nachrichtenproduzent sendet Nachrichten über Ereignisse an die registrierten Konsumenten (*Publish*)



2 wichtige Standards:

- **WS-Eventing**: BEA, IBM, MS, Sun u.a., einfache Spezifikation
- **WS-Notification**: OASIS-Standard, aktuelle Version 1.3, von IBM, SAP, Tibco, HP u.a. entwickelt, deutlich komplexer

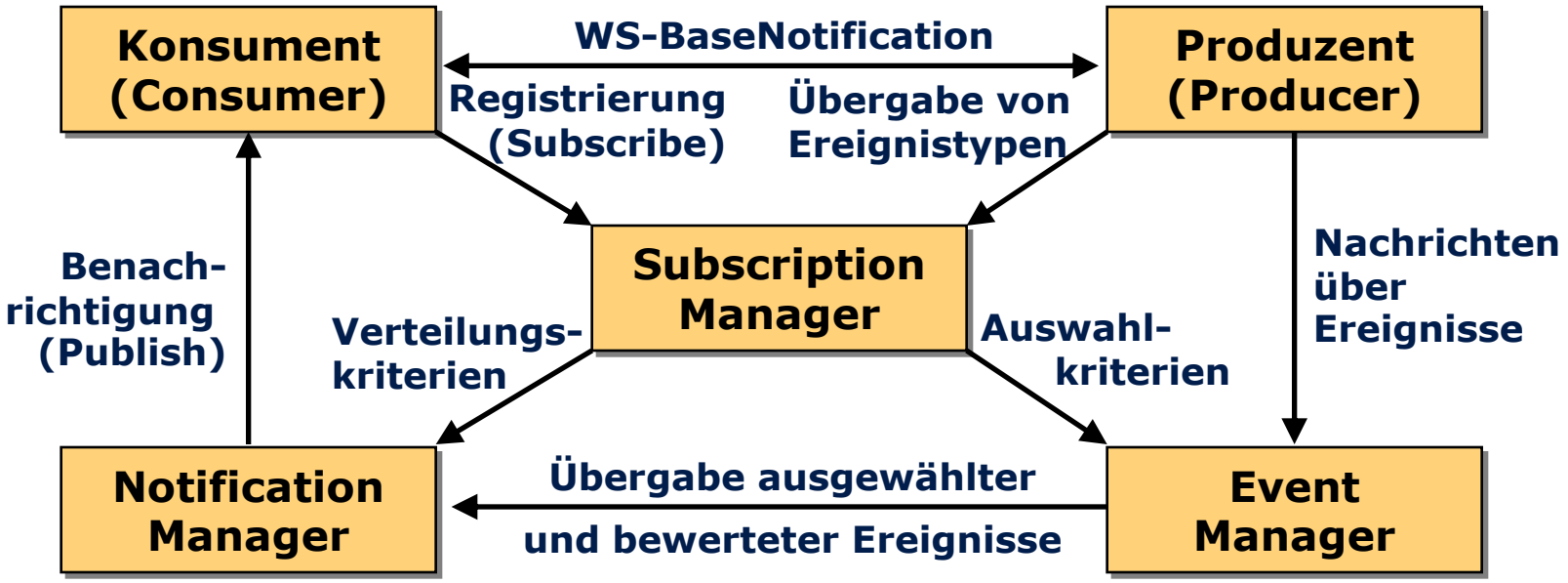
WS-Eventing: Beispiel

```
<SOAP:Header>
<wsa:Action>
   http://schemas.xmlsoap.org/ws/2004/01/eventing/Subscribe
</wsa:Action>

<wsa:ReplyTo>
   <wsa:Address>
     http://api.reisebuero.com/HotelStatusSink
   </wsa:Address>
</wsa:ReplyTo>

<wsa:To>
   http://api.holtin.com/HotelStatusSource
</wsa:To>
</SOAP:Header>
```

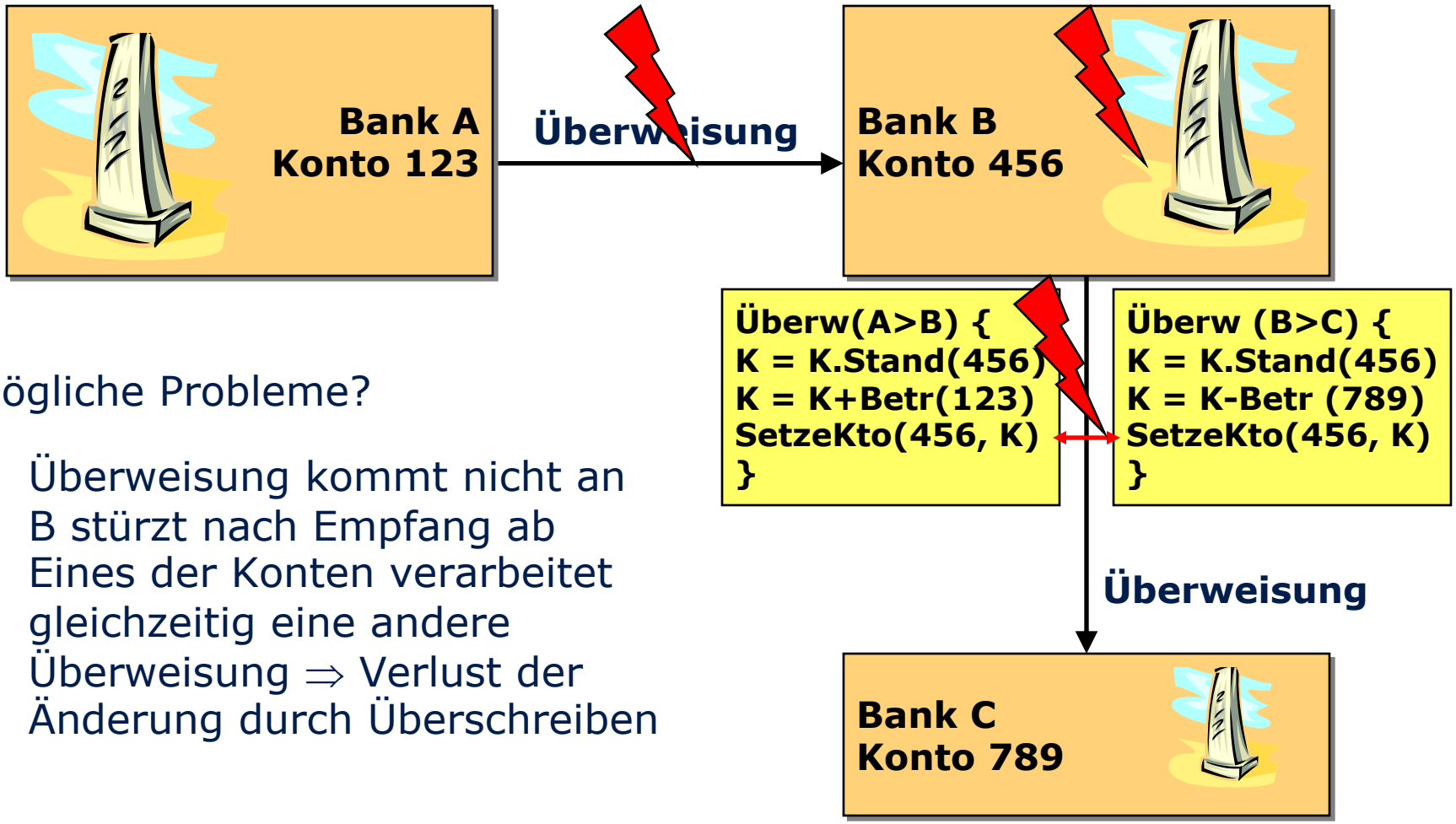
WS-Notification

Familie von 3 Spezifikationen:

- WS-BaseNotification: Producer & Consumer, Point-to-Point-Not.
- WS-Topics: Kategorisierung und Organisation von Informationen
- WS-BrokeredNotification: Schnittstelle des „Managers“



Transaktionen



Mögliche Probleme?

- Überweisung kommt nicht an
- B stürzt nach Empfang ab
- Eines der Konten verarbeitet gleichzeitig eine andere Überweisung ⇒ Verlust der Änderung durch Überschreiben

Transaktionen

- Nicht nur typisches Banküberweisungsszenario, z.B. auch:
  - Terminvereinbarungen
  - Reiseplanung

- Eigenschaften von Transaktionen:
  - Beteiligung mehrerer Teilnehmer
  - Vorher festgelegte Prozedur für jeden Teilnehmer
  - Transaktion hat bestimmtes Ergebnis
  - Ende: Teilnehmer kennen Ergebnis, stimmen diesem ggf. zu

- Definition: Eine Transaktion ist eine Menge von Arbeitsschritten mit einem konsistenten Ergebnis.

Muss ein WS Transaktionen unterstützen?

Buchung

Bestätigung: Zimmernr.

BEIGENSTERGER HOTEL Hotelbuchung

hier nicht notwendig

Buchung

Bestätigung

BEIGENSTERGER HOTEL Hotelbuchung

Eucoprar Mietwagen

hier notwendig

Web-Service-Transaktionen

Ziel:

- Interoperabilität zwischen Ausführungsumgebungen
- Verteilte Transaktionsunterstützung
- *Keine Implementierung* des Transaktionsparadigmas bei den einzelnen Teilnehmern

⇒ Nur im Umfeld von Technologien nützlich, die die Anforderungen der Protokolle implementieren

Einfluss von Transportprotokollen:

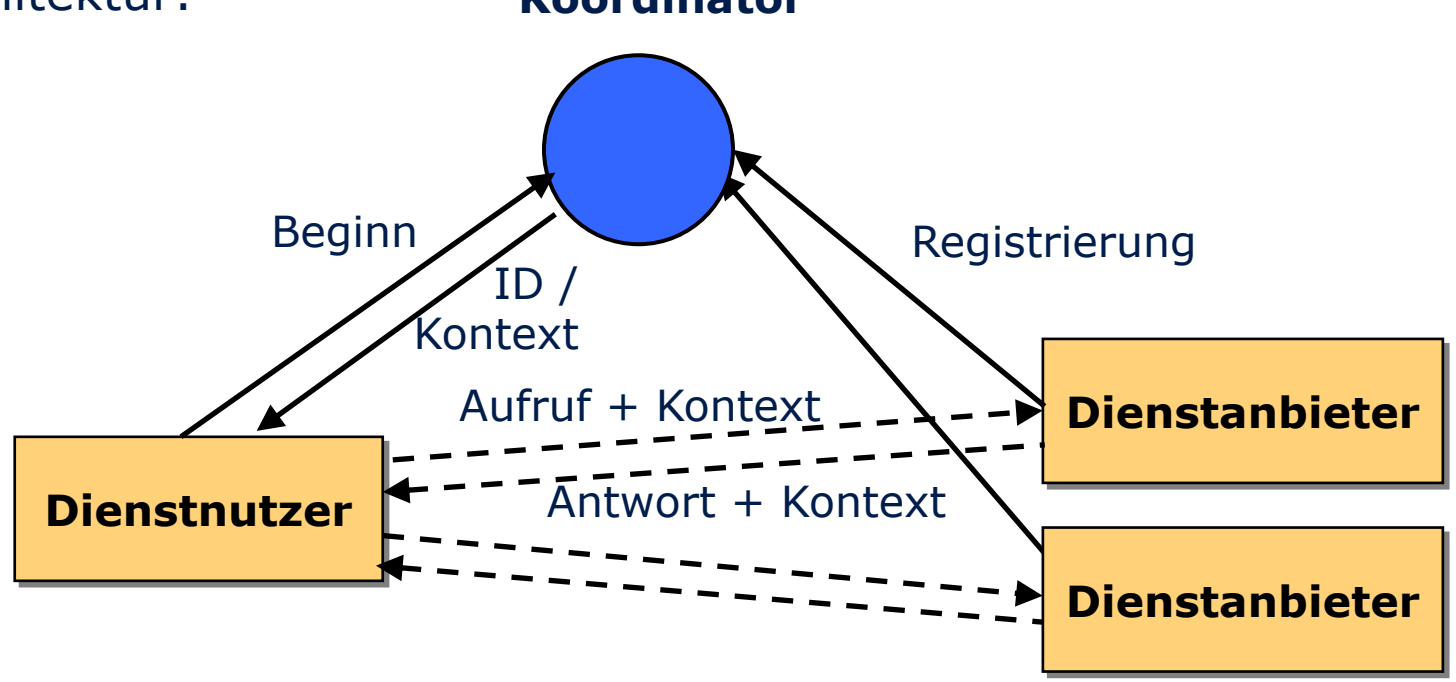
- HTTP: Nur einmaliger Request/Response-Nachrichtenaustausch

⇒ Transaktionskontext muss bei jedem Request gesendet werden

⇒ Kontext-Management für asynchrone Protokolle notwendig

Koordination

- Ziel: alle beteiligten Dienste verwenden gleichen Transaktions-Kontext
- Zentraler WS-Transaktionskoordinator: generisch
  - Verwendung verschiedener Protokoll- und Kontexttypen
- Architektur:



Transaktions-Protokolle

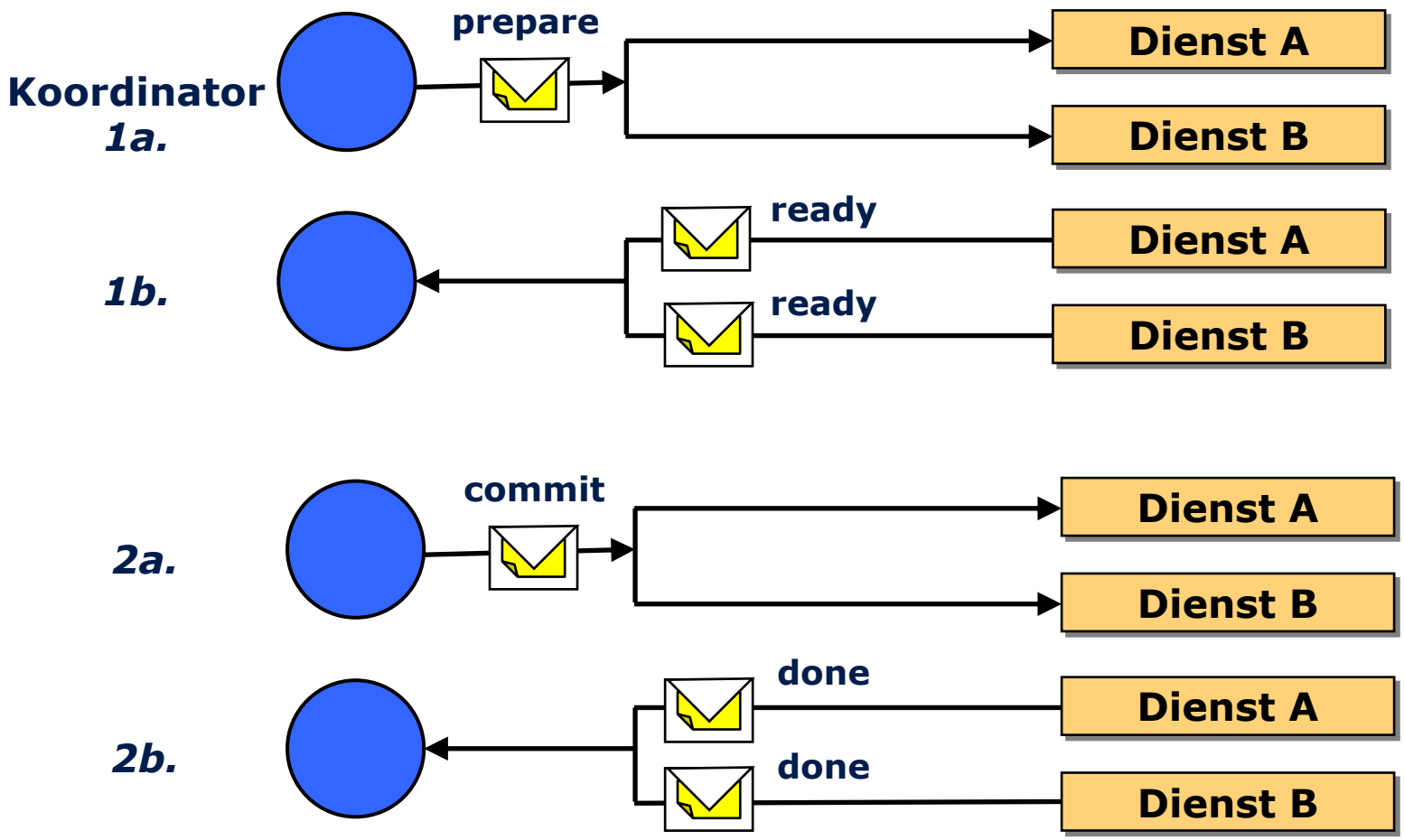
- Verschiedene Anforderungen und Umgebungen
  ⇒ Unterschiedliche Protokolle

- Für WS-Transaktionen entwickelte Protokolle:
  - **Kontextmanagement:** generisch, trennt Kontext von Transaktionsprotokollen
  - **Atomare Transaktionen:** klassisches 2-Phasen-Commit-Protokoll, stellt ACID-Eigenschaften sicher
    - Beispiel: WS-AtomicTransactions
  - **Kompensationsbasierte / langlebige Transaktionen:** ermöglichen zwischenzeitliche Commits, die über Kompensation anstatt Rollback rückgängig gemacht werden können
    - Beispiel: WS-BusinessActivity
  - BPM: Überbrückung anderer Protokolle für Transaktions-Koordination in Geschäftsprozessen
    - Bsp.: Business Process in WS-TransactionManagement

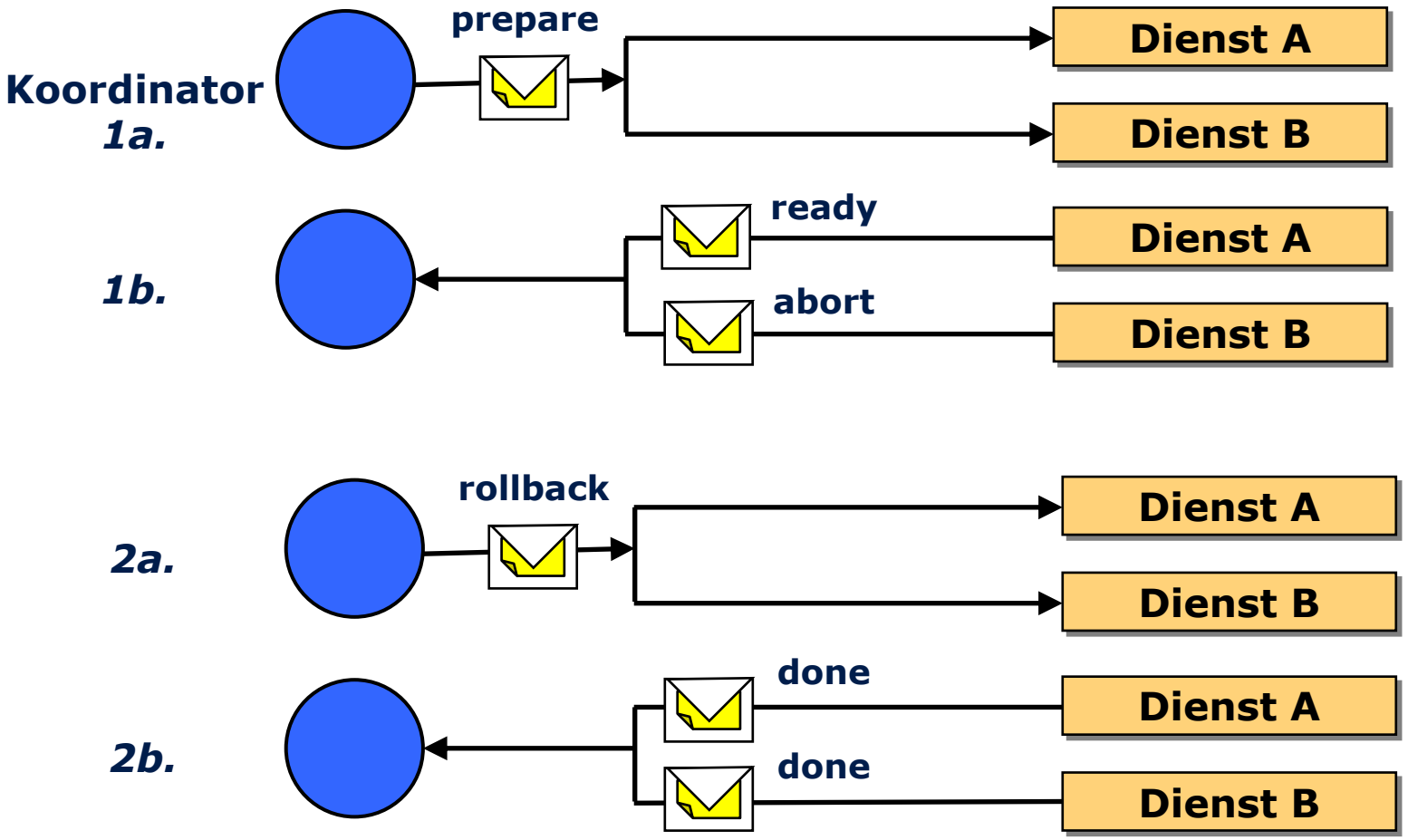
Atomare Transaktionen: ACID-Paradigma

- **Atomarität**: Alles-oder-Nichts-Prinzip, eine Menge aller Arbeitsschritte einer Transaktion wird entweder vollständig oder gar nicht ausgeführt
- **Konsistenz** (**C**onsistency): Informationssystem wird von einer Transaktion von einem in einen anderen konsistenten Zustand überführt
- **Isolation**: Durchführung einer Transaktion erfolgt isoliert von anderen gleichzeitig ausgeführten Transaktionen (T. ist völlig unbeeinflusst von anderen parallel arbeitenden T.s)
- **Persistenz** (**D**urability): bei Ende der Transaktion dauerhafte Speicherung des Transaktionsergebnisses, z.B. auf Festplatte

Atomare Transaktionen - 2PC

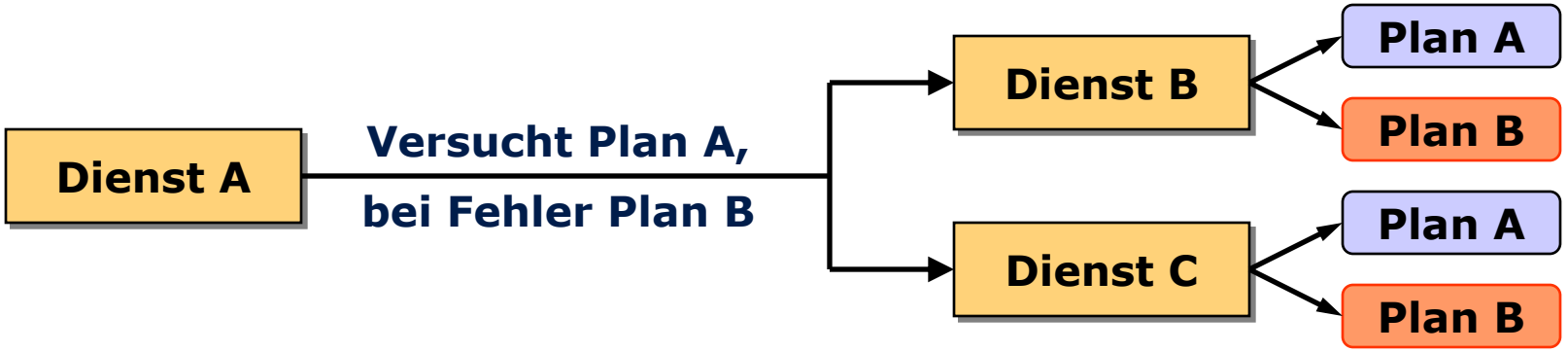


Atomare Transaktionen - Fehlerfall



Kompensationsbasierte Transaktionen

- Im täglichen Leben oft anzutreffen (Beispiel: Rückgabe eines Produktes, Kompensation: Geldrückgabe und Hinzufügen des Produktes zum Geschäftsbestand)
- Nachteile:
  - Rekursive Kompensation nicht möglich
    - Fehler bei der Kompensation – Fehlerbehandlung?
  - Kompensation für manche Operationen nicht möglich
    - Beispiel: Raketenabschuss, Ausdruck eines Dokuments



Transaktionen: Policies

- Aufgabe: Beschreibung von Transaktionsvoraussetzungen
- Ziel: Interoperabilität
- Antworten auf Fragen wie
  - Setzt ein Web Service immer einen Transaktionskontext voraus?
  - Wird ein Transaktionskontext optional unterstützt?
  - Werden überhaupt Transaktionen unterstützt?

- Beispiel: WS-AtomicTransaction-Protokoll muss vom Dienstnutzer unterstützt werden:

```
<wsp:SpecVersion
   wsp:URI="http://schemas.xmlsoap.org/ws/2003/09/wsat"
   wsp:Usage="wsp:Required"/>
```

WS-Transaktions-Spezifikationen

- **WS-Transactions**: BEA, IBM, Microsoft (2002)
  - WS-AtomicTransactions (WS-AT)
  - WS-BusinessActivity (WS-BA)
  - WS-Coordination (WS-C)
- **WS-Composite Application Framework** (WS-CAF): Oracle, Sun, Fujitsu (2005)
  - WS-Context (WS-CTX)
  - WS-CoordinationFramework (WS-CF)
  - WS-TransactionManagement (WS-TXM)

- *WS-C* und *WS-CF:* Koordinationsframeworks, die WS-AT und WS-BA unterstützen
  - WS-CF unterstützt zusätzlich die WS-TXM-Protokolle
  - WS-C dient auch zur Kontextverwaltung

WS-Coordination

- Web-Service-Konversation: Abfolge des Nachrichtenaustauschs zwischen Dienstnutzer und Anbieter eines bestimmten WS
- Web-Service-Koordinationsprotokoll: spezifiziert eine bestimmte Menge korrekter und akzeptierter WS-Konversationen
- Koordination: 3 Phasen
  - Aktivierung (Erstellung des Transaktionskontextes)
  - Registrierung (Web Services bestätigen ihre Teilnahme an der Transaktion)
  - Koordination (der Ergebnisse der teilnehmenden WS)

WS-AtomicTransaction

Geeignet für

- Kurzzeitige Transaktionen
- Transaktionen innerhalb einer Anwendung

- WS-AT unterstützt 2 Versionen von 2PC (2-Phasen-Commit):
  - volatile (flüchtig, d.h. Operationen auf RAM) und
  - durable (persistente Speicherung, z.B. auf Festplatte)
- Ziel dieser Unterscheidung: Koordination von Operationen auf flüchtigem Speicher vor persistenten Operationen

WS-BusinessActivity

Geeignet für

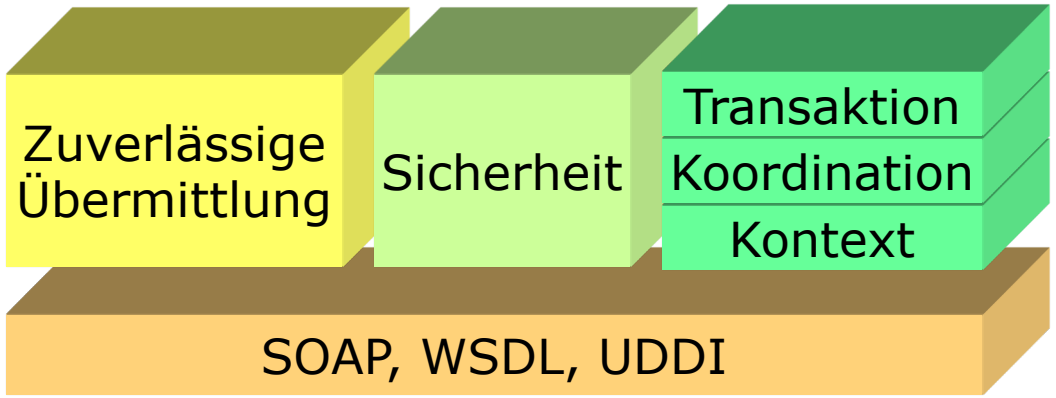
- Langandauernde, verteilte Anwendungen, die viele Ressourcen und atomare Transaktionen verwenden könnten
- Anwendungen, die bei Bedarf die Transaktion auch ohne Koordinator beenden müssen

- Erlaubt sowohl ACID-Transaktionen als auch Kompensation
- Definition von Business-Logik für Ausnahmefälle möglich
- Kann als Wrapper für existierende, proprietäre Workflow- und Business-Process-Systeme dienen $\Rightarrow$ Interoperabilität

Zusammenfassung

- Zuverlässige Übermittlung:
  - WS-Addressing, WS-MessageDelivery
  - WS-ReliableMessaging, WS-Reliability
- Benachrichtigung: WS-Eventing, WS-Notification
- Transaktion, Koordination, Kontext:
  - WS-Transactions (Coordination, AtomicTransaction, BA)
  - WS-CAF (Context, CoordinationFramework, TransactionM.)

„WS-* Erweiterungen“



Web Services

TECHNISCHE UNIVERSITÄT DRESDEN

Vorlesung "Service and Cloud Computing"

6. SaaS - Suche von Diensten , WS-Erweiterungen, SOA-Infrastruktur

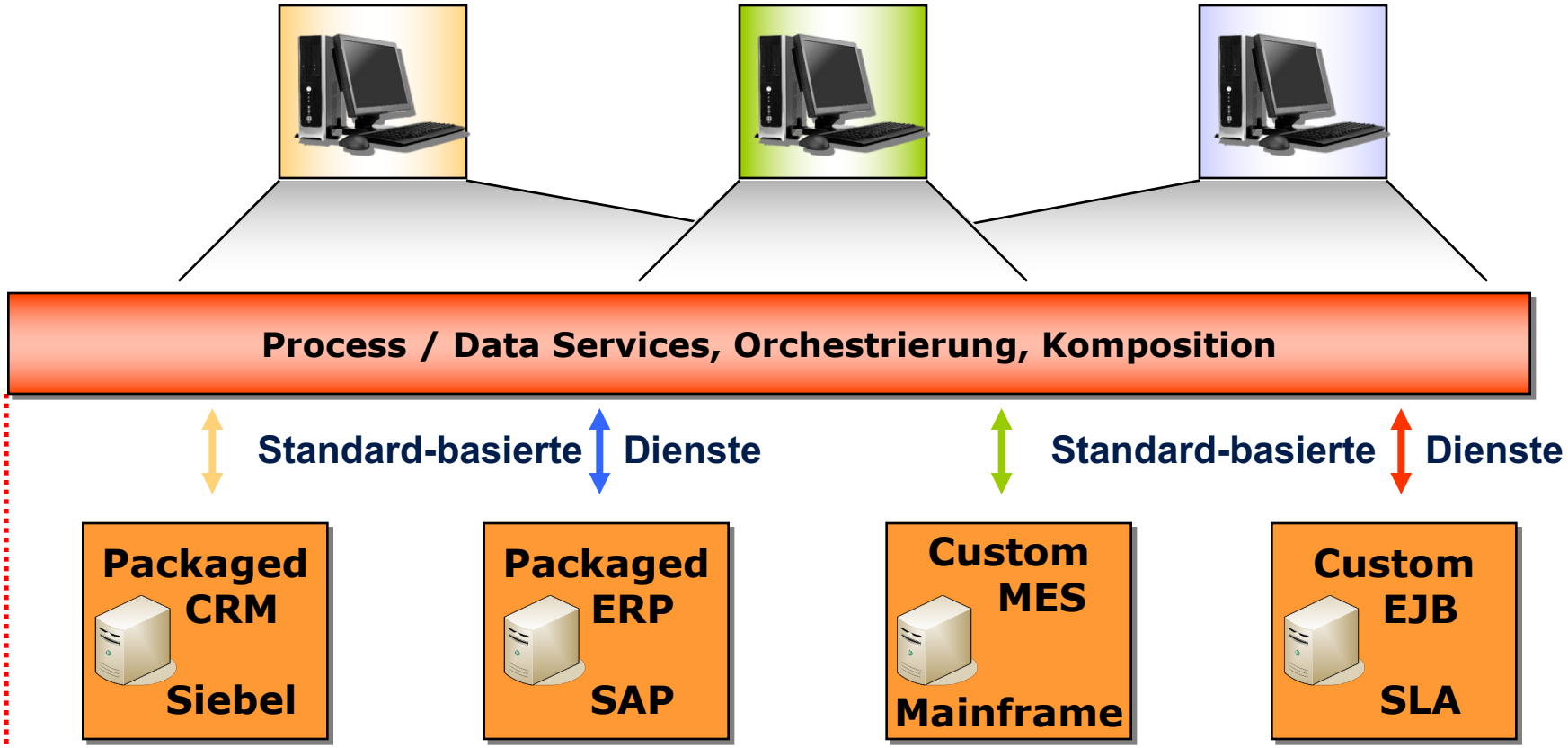
6.2 SOA-Infrastruktur

Dr.-Ing. Iris Braun

Gliederung

- WS-* Erweiterungen
  - Adressierung
  - Zuverlässige Nachrichtenübertragung
  - Asynchrone Nachrichtenübertragung
  - Transaktionen
  - Festlegung von Richtlinien zur Dienstnutzung
- Frameworks zur technischen Umsetzung einer SOA
  - ESB – Enterprise Service Bus
  - JBI – Java Business Integration
  - SCA – Service Component Architecture

Technische Umsetzung einer SOA: Überblick



Wie sollte die Umsetzung aussehen? Zusätzliche Dienste?

Enterprise Service Bus

- Infrastruktur innerhalb einer SOA, über die Dienstanbieter und Dienstnutzer miteinander kommunizieren

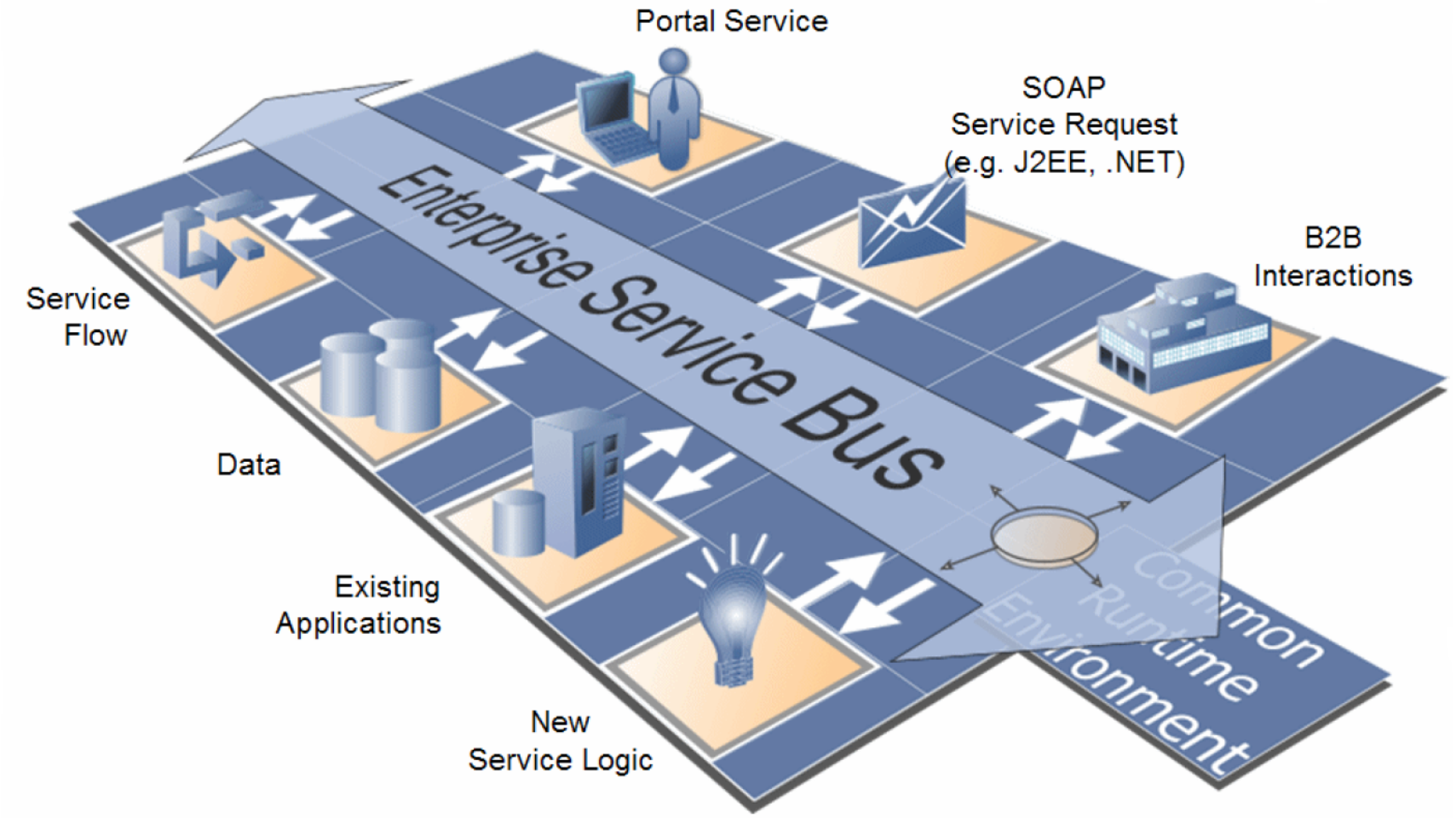
zentrale Eigenschaften eines ESB [Gartner Group]:

- Kommunikation via Message-oriented Middleware (MOM) z.B. JMS (Java Messaging Service)
- Konnektivität auf Basis von Web Services (SOAP, WSDL, REST)
- Transformation von XML- und SOAP-Nachrichten inklusive Routing
- stellt weitere Fähigkeiten wie Sicherheits-, Single-Sign-On-, Registry- und Datenkonvertierungsdienste zur Verfügung

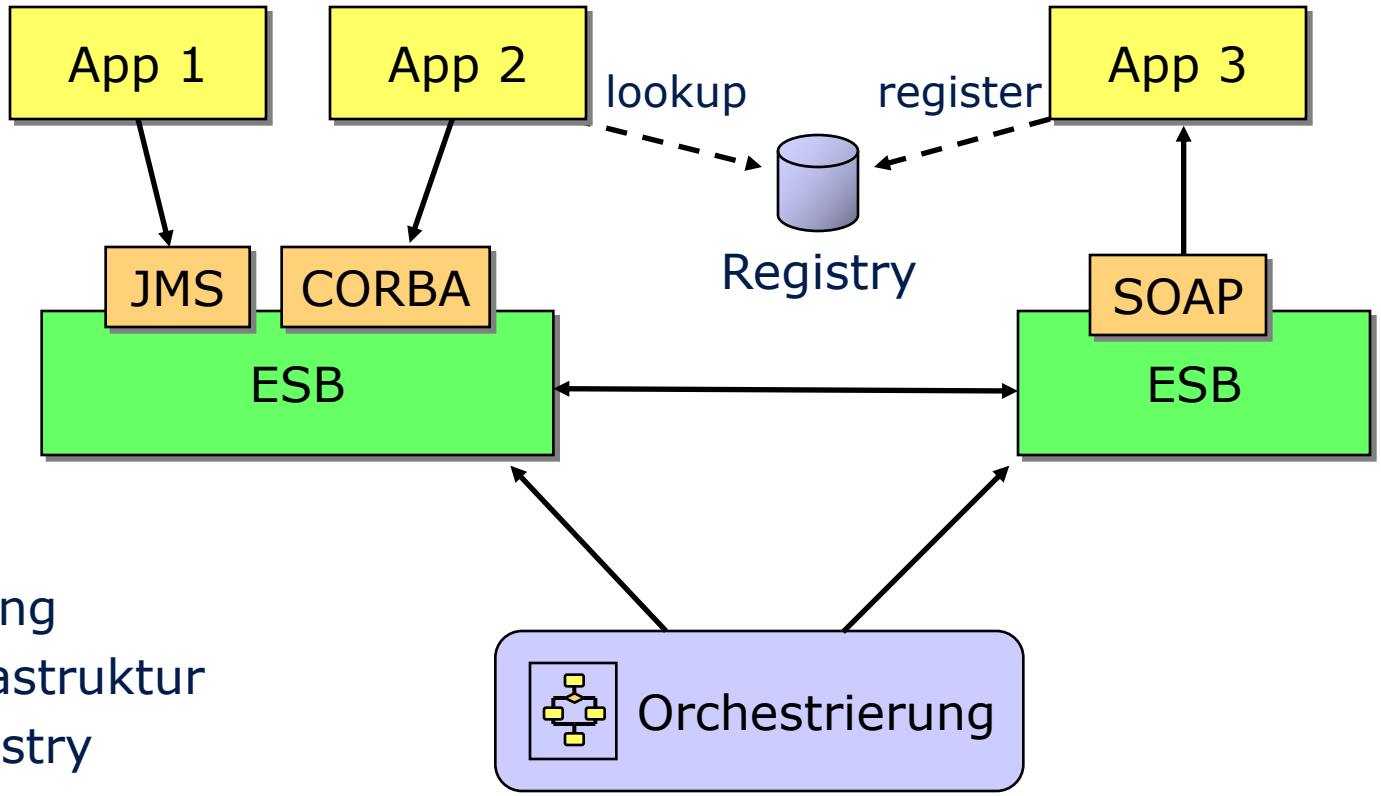
Anforderungen:

- Unterstützung vieler Standardprotokolle = Integrationsfähigkeit
- Zuverlässiger Nachrichtentransport (über Messaging Middleware)
- Zentrale Überwachung, Monitoring
- weitere Features wie Skalierbarkeit, Transaktionsunterstützung, inhaltsbasiertes Routing, Adapter für alle gängigen Nachrichten-formate

Enterprise Service Bus: Überblick



Enterprise Service Bus: Beispiel



- Verteilung
  - Infrastruktur
  - Registry
- Komponentenarchitektur

JBI: Java Business Integration

Begriff Enterprise Service Bus nicht standardisiert

⇒ Angebote verschiedener Hersteller nicht kompatibel

Java Business Integration: JSR-208

Ziele:

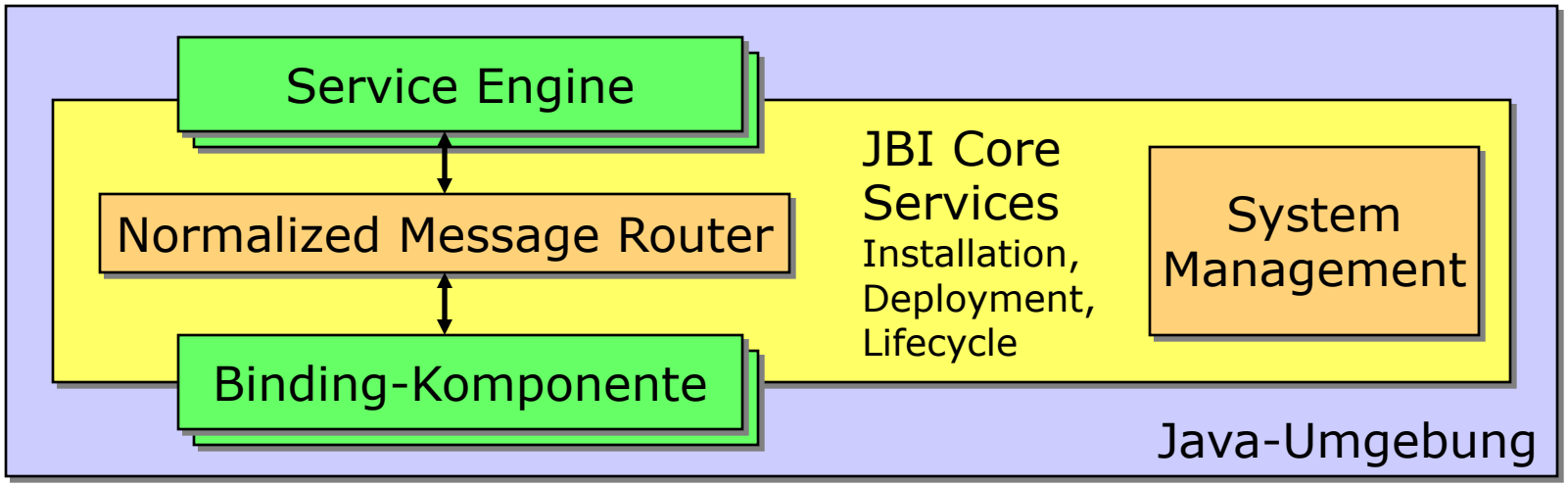
- Integrationsfähigkeit für Java
- Entwicklung eines Industriestandards
- Herstellerunabhängigkeit
- Aufbauend auf vorhandenen Standards

- Entwickelt durch viele Firmen, außer IBM und BEA

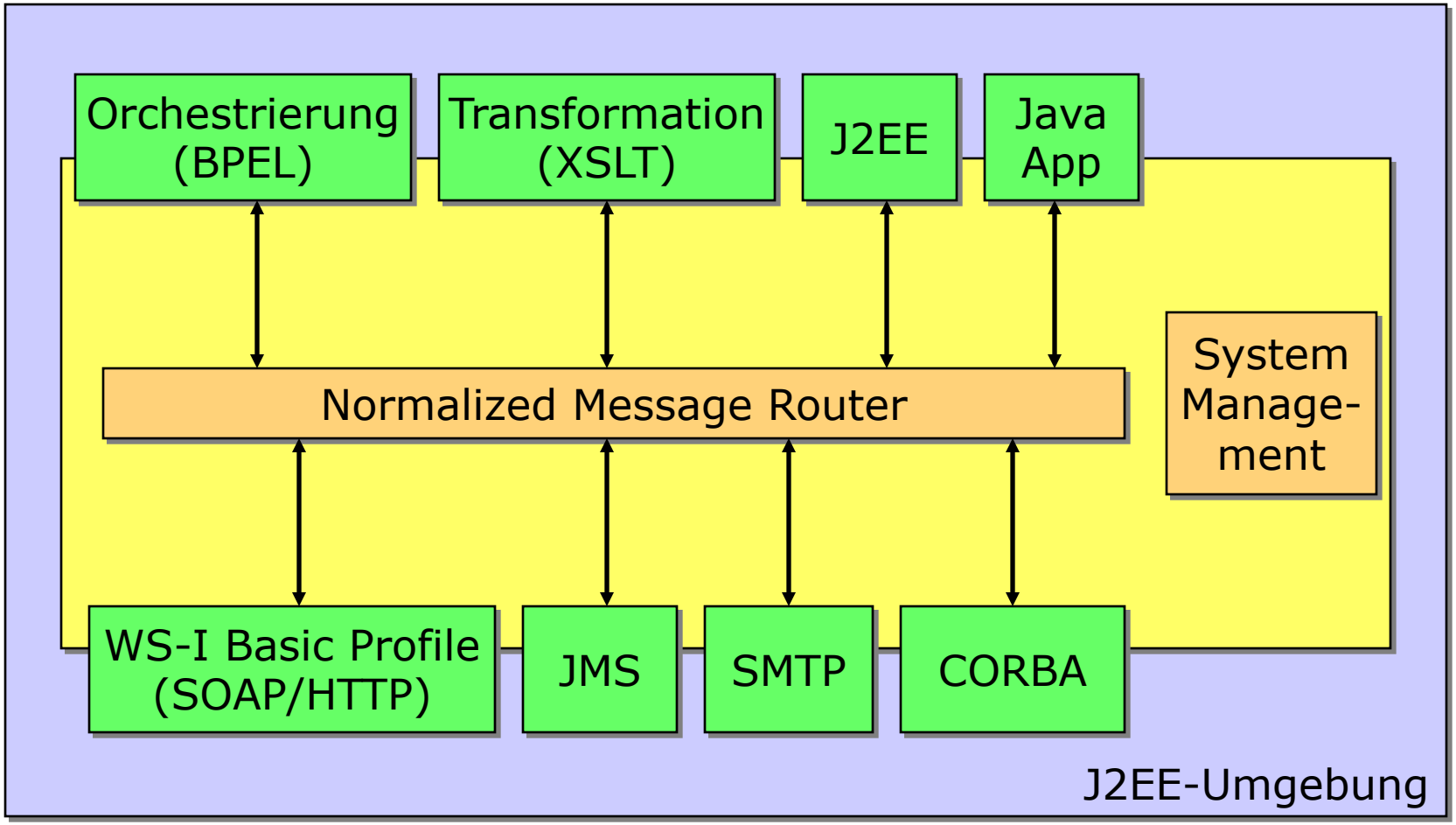
JBI: Architektur

JBI-Konzept lässt sich unterteilen in

- **Service Engines**: erweiterbare Geschäftslogik, z.B. EJB-Wrapper
- **Binding-Komponenten**: Proxy für Dienstnutzer und entfernte Dienste ⇒ transportprotokoll-unabhängiger Zugriff
- **Normalized Message Router**: Versand und Routing von Nachrichten über einen Delivery Channel
- **JBI Laufzeitumgebung**

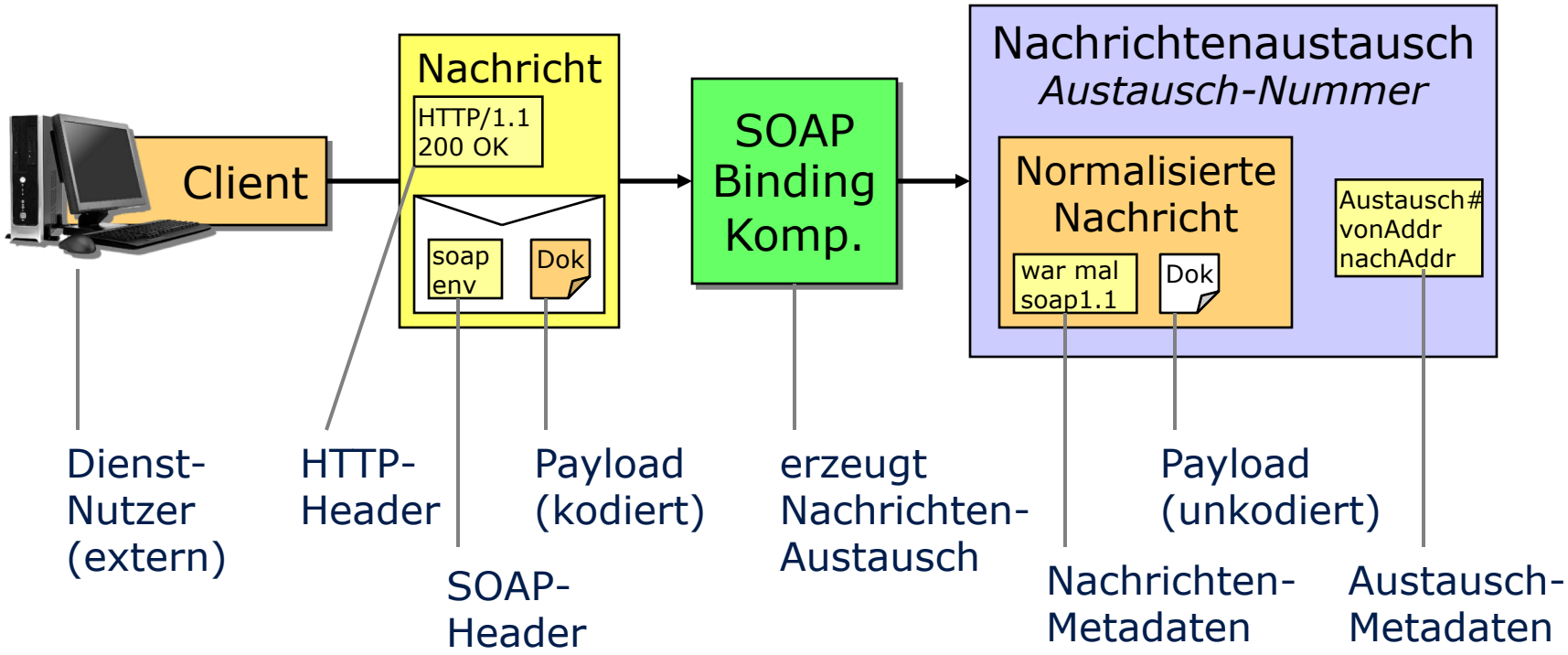


JBI: Beispiel



JBI: Nachrichtenaustausch

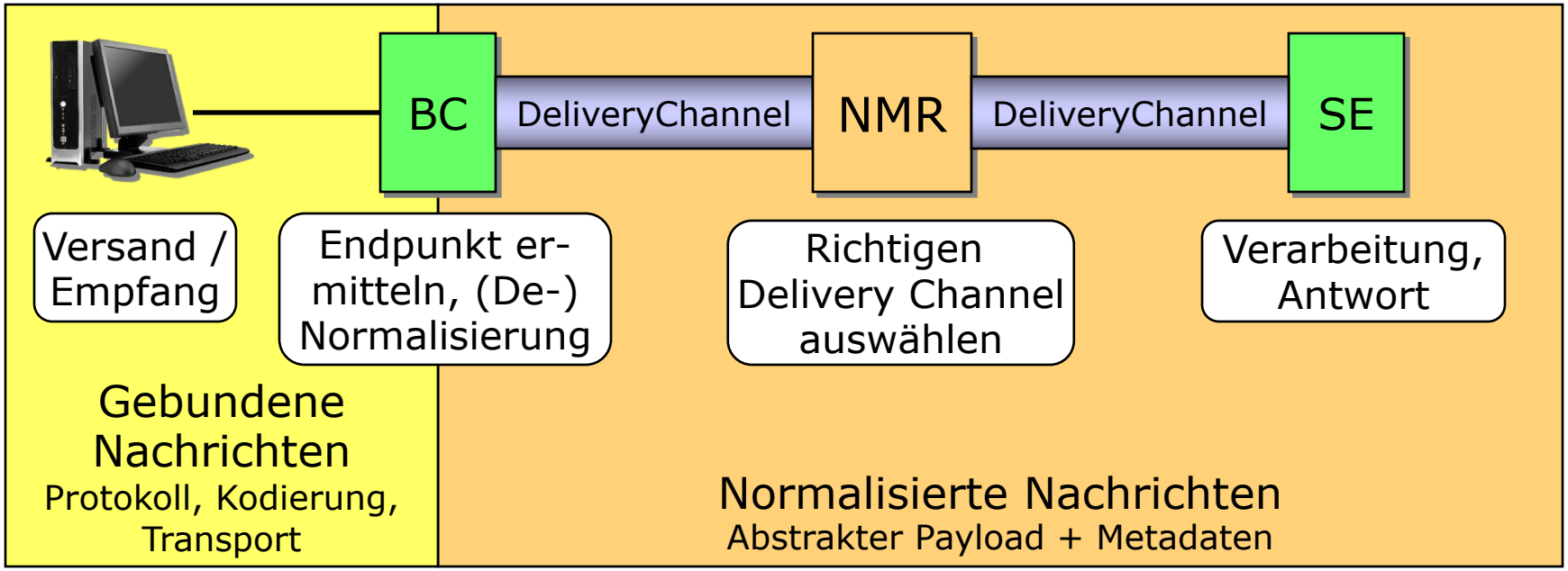
- Umwandlung der protokollspezifischen Nachrichten in unkodierte Dokumente und zugehörige Metainformationen wie protokoll-spezifischer Kontext, Transaktionsinformationen usw.



JBI: Normalized Message Router

Aufgaben des Normalized Message Router (NMR):

- Interoperabilität der Komponenten
- Routing der Nachrichten zu den passenden Endpunkten
- Austausch normalisierter Nachrichten (Payload + Metadaten)



SCA: Service Component Architecture

Ziel der **Service Component Architecture (SCA)**:

- Integration verschiedener serviceorientierter Architekturen
- Reduzierung des Aufwands bei der Entwicklung der Infrastruktur

Vergleich mit JBI:

- Nicht Java-spezifisches, metadaten-basiertes Modell zur Komposition von Diensten

Verteilte objektorientierte Anwendung	Verteilte serviceorientierte Anwendung
Eng gekoppelte Sammlung aus Objektkomponenten, Interaktion über technologiespezifische Protokolle	Lose gekoppelte Zusammensetzung aus Servicekomponenten, technologieunabhängige Interaktion

SCA: Überblick

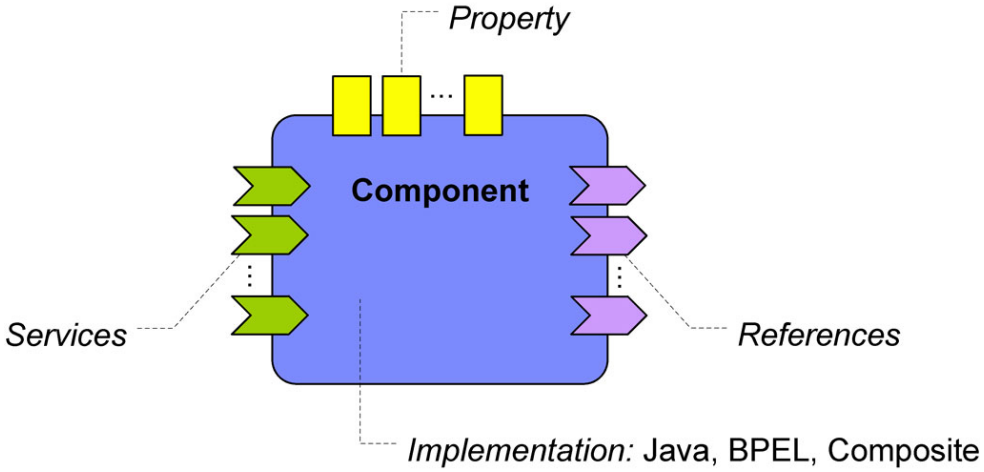
- Erster Entwurf im November 2005, Veröffentlichung durch die Open SOA Collaboration (osoa.org) im März 2007
- Zur Standardisierung an OASIS übergeben

SCA definiert Syntax und Semantik für

- Konstruktion von Diensten (Implementation)
- Verbindung von Komponenten zu einem Prozess (Assembly)
- Deployment in ein Gesamtsystem (Activation)

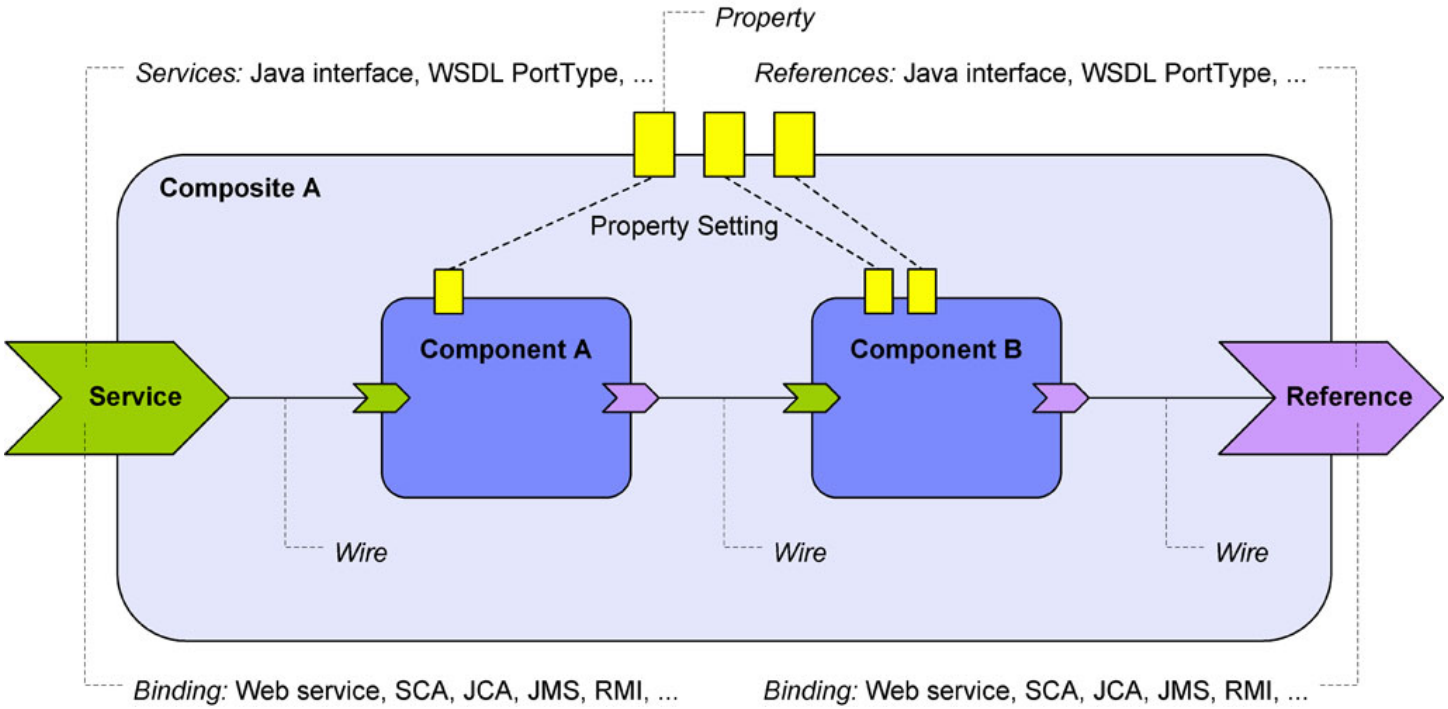
SCA: Komponenten

- Komponenten repräsentieren Business-Funktionen
- Konfigurierte Instanz einer Implementation (beliebige Sprache)
- Funktion wird anderen Komponenten über Services angeboten
- Abhängigkeiten von anderen Komponenten über References
- Properties: Eigenschaften, die die Ausführung der Komponente beeinflussen können

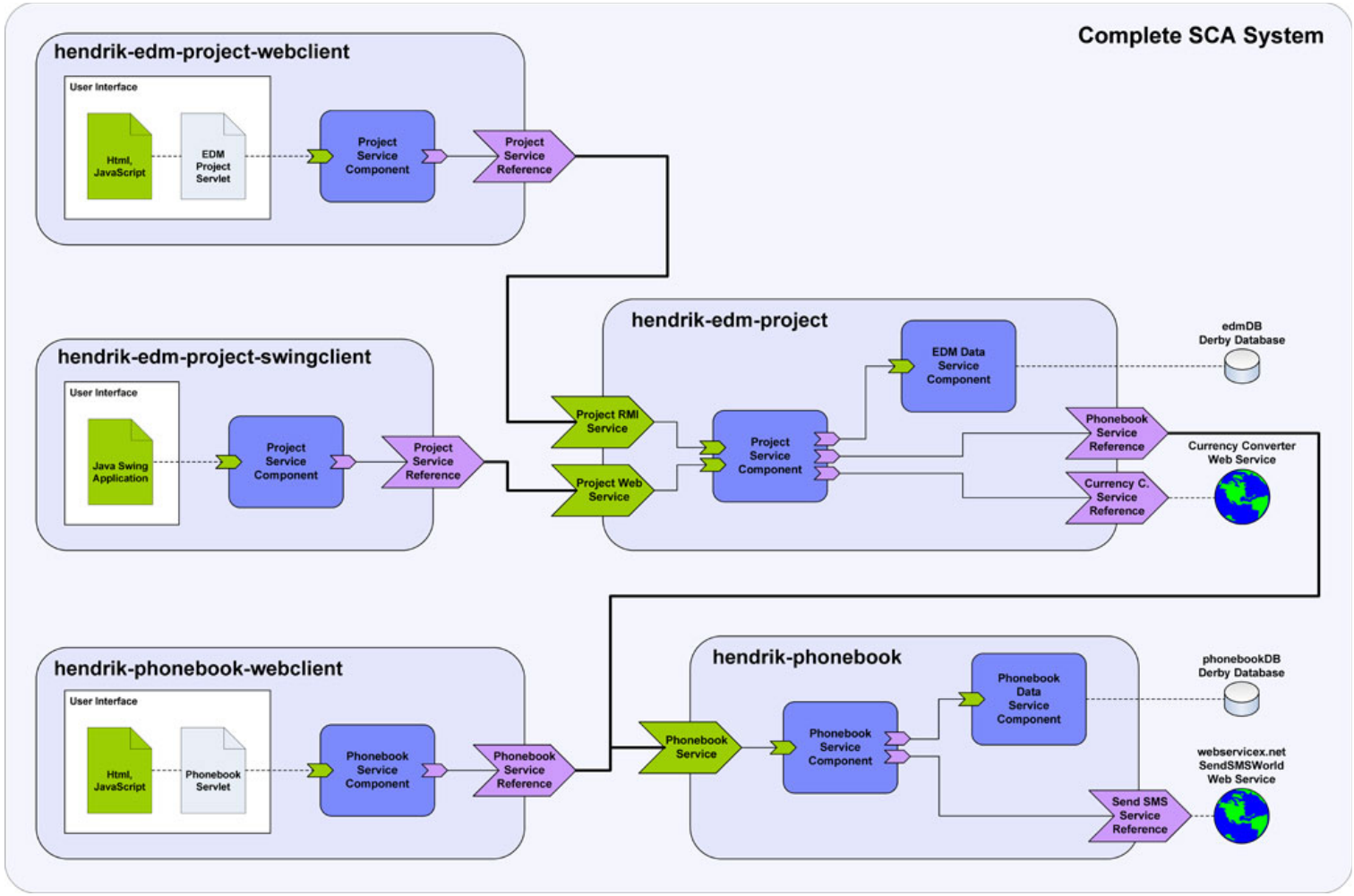


SCA: Kompositionen

- Fest verkoppelte Zusammensetzung von Komponenten, Referenzen, Diensten, Verbindungen und Eigenschaften
- Komponenten über verschiedene Technologien implementierbar
- XML-Konfigurationsdatei enthält strukturelle Details



SCA: Beispiel für ein vollständiges System



SCA: Vorteile und Nachteile

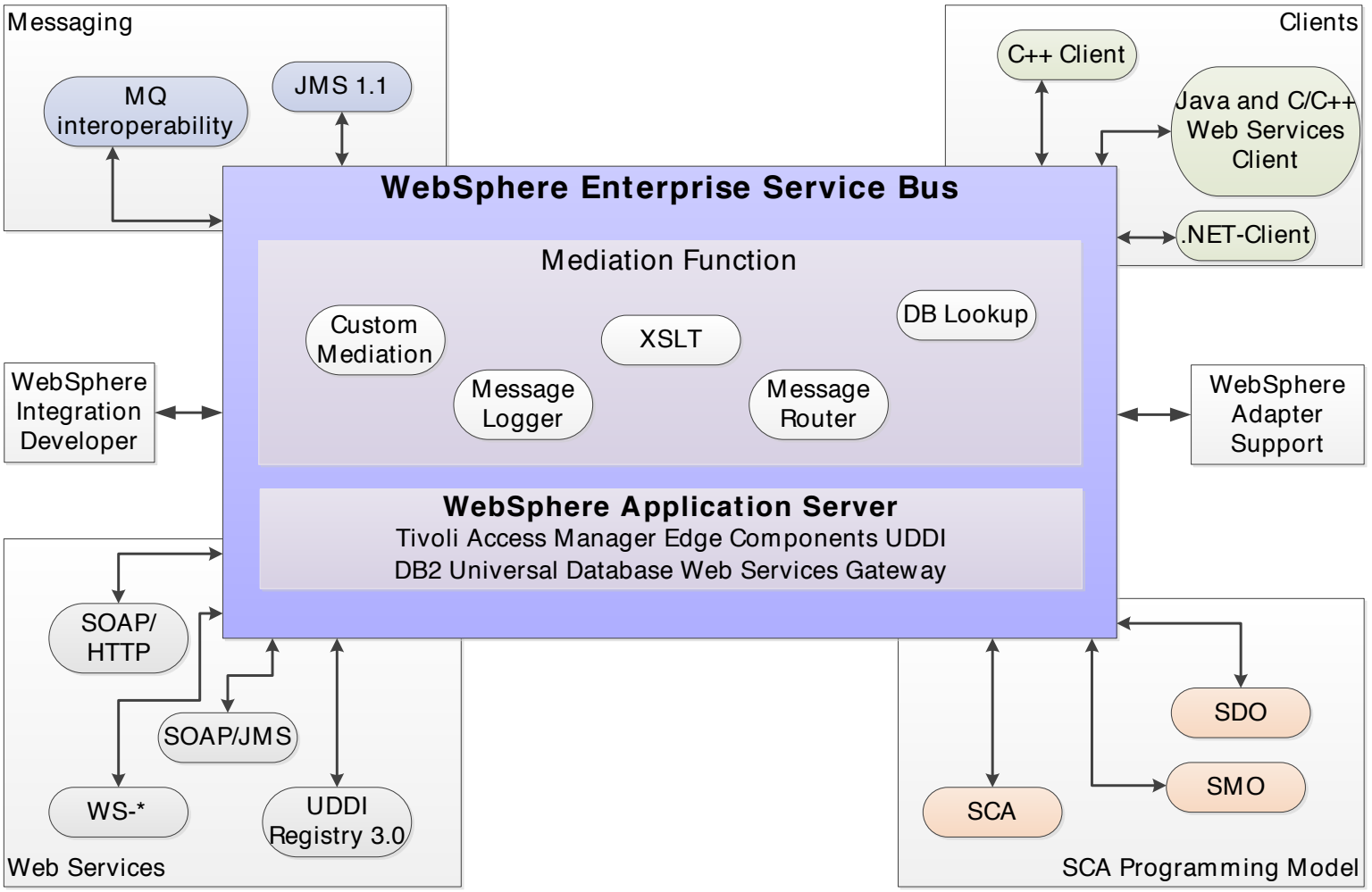
Vorteile:

- Trennung von Komponenten-Implementierung und Dienstnutzung
- Flexibilität und Wiederverwendbarkeit von Kompositionen
- Unabhängigkeit von Implementierungstechnologien
- Kein Vendor-Lockin (Abhängigkeit von einem Anbieter)
- Einfachere und schnellere Anwendungsentwicklung

Nachteile:

- Keine Wiederverwendbarkeit von Kompositionsbestandteilen
- Nicht ausreichende Quality-of-Service-Unterstützung
- Apache-Implementierung „Tuscany“ noch unzuverlässig und schlecht dokumentiert

ESB Produkt Beispiel: IBM WebSphere ESB



Zusammenfassung SOA-Infrastruktur

Technische Umsetzung einer SOA:

- Infrastruktur zum Nachrichtenaustausch, zur Dienstkomposition und Dienstverwaltung
- Beispiele:
  - ESB - Enterprise Service Bus,
  - JBI – Java Business Integration,
  - SCA – Service Component Architecture

- Routing von Serviceanfragen und Auflösung von Versionskonflikten
- Datentransformation und –mapping (Mediation)
- Service-Orchestrierung, Prozessmanagement, Transaction Management
- Sicherheit, Quality of Service, SLAs
- Service Registry und Metadatenverwaltung
- Monitoring und Management