Abschlussvortrag zur Diplomarbeit Dienstkomposition mit Hilfe semantischer Service-Templates

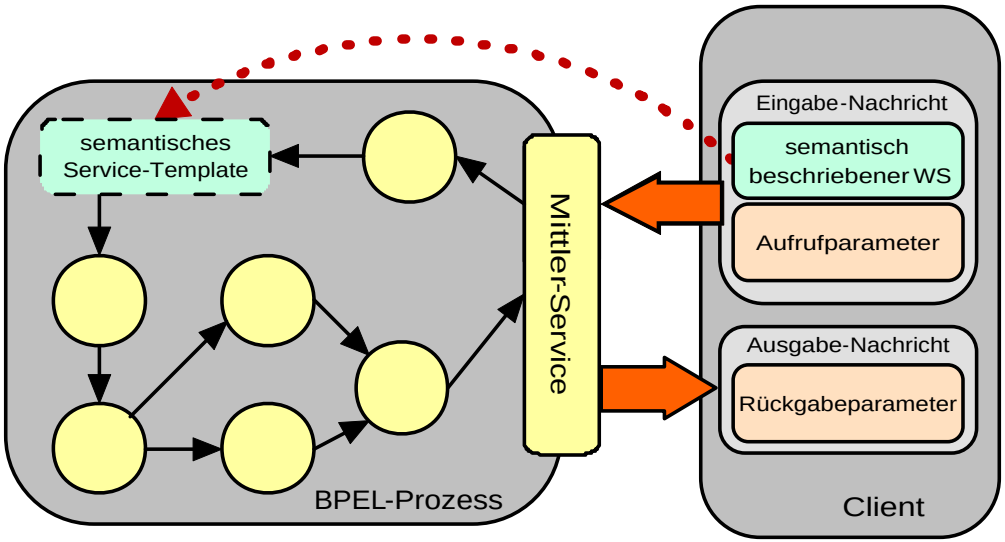
Tihomir Magdic

Motivation

- Standards zur Komposition von Diensten sind beschränkt auf deren syntaktische Struktur
- Die Funktionalität eines Dienstes wird bei der Komposition nicht berücksichtigt
- Einfluss eines Clients auf die Ausführung einer Komposition ist gering
  $\Rightarrow$ beschränkt sich auf die Übergabe von Parametern

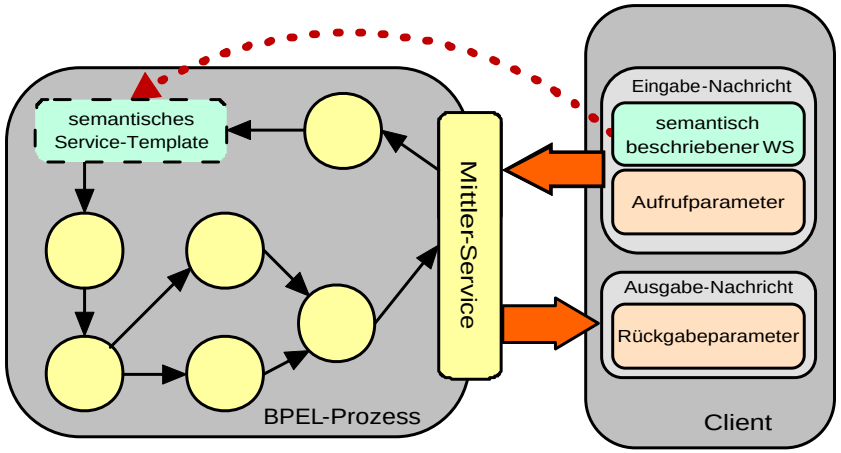
Mittler-Szenario

- gekennzeichnet durch die Flexibilisierung der Dienstkomposition
- Client hat die Möglichkeit Dienste, die ausgeführt werden, direkt zu bestimmen
- Service-Template: beschreibt die Funktionalität eines Dienstes



Mittler-Service

- bietet einen Dienst an, für dessen Erbringung er andere Dienste benötigt
- enthält mindestens ein semantisches Service-Template
- enthaltene Service-Templates werden veröffentlicht
- jedem semantischen Service-Template kann ein Client einen konkreten Dienst zuweisen



Ziele der Arbeit

- Entwicklung und prototypische Implementierung einer Architektur für das Mittler-Szenario
- Erweiterung von BPEL: Einbindung semantischer Service-Templates
- einfache Integration
  - vorhandene Dienste und Dienstkompositionen sollen an das Mittler-Szenario angepasst werden können

Was muss betrachtet werden?

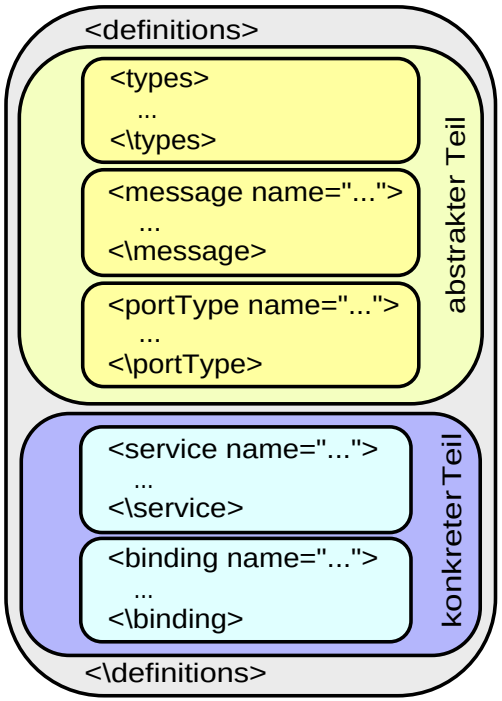
- WS-Standards für Service-Schnittstellen und -Komposition: WSDL, WS-BPEL
- Wie kann die Funktionalität eines Dienstes beschrieben werden?

WSDL – Web Service Description Language

- Standard zum Beschreiben von Web-Service-Schnittstellen.
- Aktuelle Version ist WSDL 2.0. Da WS-BPEL auf WSDL 1.1 aufbaut, wird diese betrachtet.
- beschreibt:
  - die abstrakte Service-Schnittstelle: angebotene Operationen, Nachrichten, Datentypen, etc.
  - wie ein Service zu benutzen ist: Netzwerk-Adresse, Protokollbindungen, Nachrichten-Encodierung

WSDL - Aufbau

- Unterscheidung zwischen abstraktem und konkretem Teil



WS-I Basic Profile (1)

- Eines der Hauptziele der WS-Standards: Überwindung von Heterogenität
- Problem: Flexibilität der Standards verringert Kompatibilität zwischen Diensten
- Weiteres Problem: Inkonsistenzen und Mehrdeutigkeiten in den Spezifikationen
- $\Rightarrow$ Zur Gewährleistung der Interoperabilität müssen Inkonsistenzen und Mehrdeutigkeiten in WS-Standards aufgelöst werden.
- $\Rightarrow$ Flexibilität muss beschränkt werden, um Interoperabilität zu gewährleisten
- *WS-I BP* ist eine Spezifikation von des Industriekonsortiums *Web Services Interoperability*.

WS-I Basic Profile (2)

- Ziel des WS-I: Interoperabilität von WS-Spezifikationen
- Stellt grundlegende Regeln für WSDL, SOAP und UDDI bereit.
- Diese sollen die Interoperabilität von Diensten gewährleisten, die diese Standards verwenden.
- Einige Regeln:
  - Nachrichtenaustauschmuster: nur one-way und request-response
  - Bindings: nur rpc-literal und document-literal
  - Transportprotokoll: nur HTTP

WS-BPEL – Service-Komposition

- WS-BPEL ermöglicht die deklarative Spezifikation von Service-Kompositionen
- aktuelle Version WS-BPEL v2.0 ist seit April 2007 *OASIS*-Standard
- Unterscheidung abstrakter vs. ausführbarer Prozess
- Für Mittler-Szenario wird die Definition ausführbarer Prozesse betrachtet

WS-BPEL und WSDL

- WS-BPEL setzt auf WSDL 1.1 auf
- verwendet in der Prozessdefinition nur den abstrakten Teil von WSDL
- Binding-Informationen müssen spätestens zur Laufzeit vorhanden
- Nachrichtenaustauschmuster in WS-BPEL: *one-way* und *request-response*

WS-BPEL – Zuweisung von Endpoint-Referenzen

- Endpoints sind konkrete Dienstimplementierungen der enthaltenen Partner-Services
- WS-BPEL erlaubt die Zuweisung von Endpoints auf unterschiedliche Arten
- Zuweisungsarten:
  - Statische Zuweisung beim Prozessentwurf
  - Statische Zuweisung beim Deployment
  - Dynamische Zuweisung als Ergebnis einer Suche
  - Dynamische Zuweisung durch einen Partner-Service

Service-Aufruf in WS-BPEL

- BPEL definiert Aktivitäten für die Spezifikation der Prozesslogik
- Aktivitäten als Kontrollstrukturen:
  - `sequence`, `if`, `while`, `repeatUntil` `flow`, etc.
- Aktivitäten für die Interaktion mit Partner-Services:
  - `receive`, `reply`, `invoke`, `pick`
- die `invoke`-Aktivität dient zum Aufrufen eines Partner-Services

WS-BPEL – Erweiterbarkeit

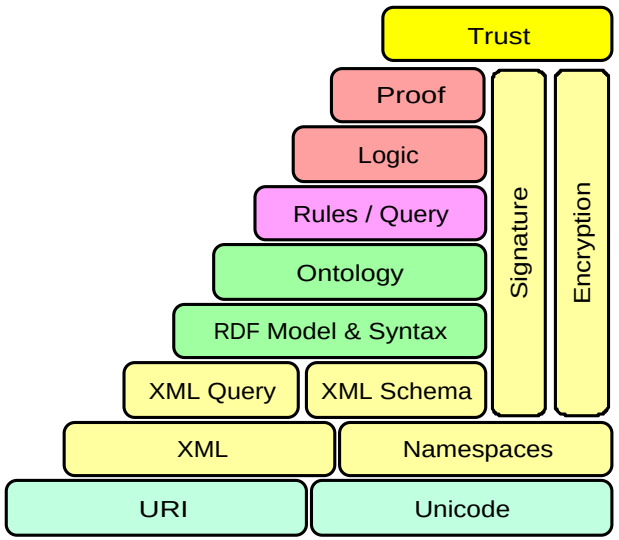
- WS-BPEL ermöglicht generelle Erweiterbarkeit
  - selbstdefinierte Attribute und Elemente können zu BPEL-Elementen hinzugefügt werden
- zusätzlich werden die beiden Erweiterungskonstrukte `<extensionAssignOperation>` und `<extensionActivity>` definiert
- `<extensionActivity>` dient zum definieren neuer Aktivitäts-Typen
  - Erweiterungsaktivitäten müssen innerhalb dieses Containers platziert werden
  - es kann angegeben werden, ob die Erweiterungen von einer BPEL-Engine verstanden werden müssen

Wie kann die Funktionalität eines Services beschrieben werden?

- die Funktionalität muss in maschinenverständlicher Form beschrieben werden
- hierfür muss Wissen explizit formalisiert werden
- *Ontologien*:
  - Begriff kommt aus der Philosophie:
    - steht für die planmäßige Darstellung von Existenz (allen Dingen, die existieren)
  - *Eine Ontologie ist eine explizite Spezifikation einer gemeinsamen Konzeptualisierung.*
  - ermöglichen es Dinge mit ihrer Bedeutung – Semantik – zu beschreiben

Semantisches Web

- der Begriff *semantic Web* wurde von Tim Berners-Lee geprägt
- dem World Wide Web fehlt es an Möglichkeiten, den Inhalt von Web-Resourcen semantisch zu beschreiben
  - $\Rightarrow$ Die Suche ausgehend von der inhaltlichen Bedeutung von Resourcen gestaltet sich schwierig
  - W3C hat ontologiebasierte Sprachen zur Wissensrepräsentation im semantischen Web vorgeschlagen



RDF – Resource Description Framework

- erste Sprache, die speziell für die Unterstützung des semantischen Web entwickelt wurde
  - ermöglicht die Beschreibung semantischer Informationen
- zugrunde liegende Struktur: *RDF-Tripel*:
  - Subjekt $\Rightarrow$ Prädikat $\Rightarrow$ Objekt
  - Verkettung: S $\Rightarrow$ P $\Rightarrow$ O $\Rightarrow$ P $\Rightarrow$ O, (S $\Rightarrow$ P $\Rightarrow$ O) $\Rightarrow$ P $\Rightarrow$ O
- definiert nur die Syntax für die semantische Beschreibung, nicht aber ihre Interpretation
  - *Das Haus läuft über die Straße* ist eine RDF-Aussage
  - $\Rightarrow$ Prädikat *Laufen* ist nicht auf das *Haus* anwendbar
    - diese Einschränkung kann nicht in RDF definiert werden
    - Sprache für die Definition solcher Regeln (Vokabulare): *RDF-Schema*

RDF-Schema und OWL

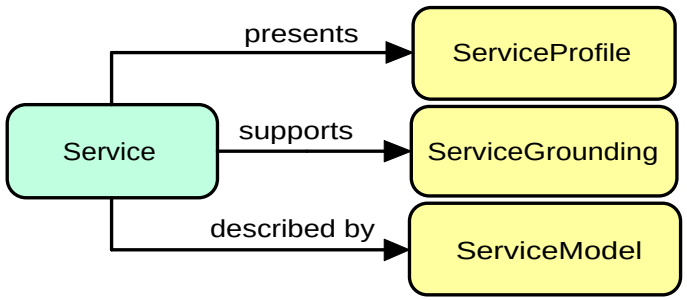
- RDF-Schema ermöglicht die Definition von Vokabularen basierend auf RDF
  - für Properties und Klassen können hierarchische Beziehungen beschrieben werden
- OWL erweitert die Beschreibungsmächtigkeit von RDF-Schema
  - bietet weitere Sprachelemente zum Beschreiben von Klassen, ihren Eigenschaften und ihren Beziehungen
  - drei OWL-Sprachvarianten:
    - *OWL-Lite*, *OWL-DL* und *OWL-Full*
    - OWL-Full ist am ausdrucksstärksten. Allerdings können Vokabulare, die mit OWL-Full erstellt sind, unentscheidbar sein.
    - OWL-DL und -Lite sind entscheidbar

Semantische Web Services – Grundlagen

- Vorgehensarten für die semantische Beschreibung von Web Services:
  - *top-down*:
    - semantische Dienstbeschreibung ist unabhängig von der Schnittstellenbeschreibung
    - $\Rightarrow$ es muss ein *Grounding* spezifiziert werden
    - das *Grounding* verknüpft die semantische Beschreibung mit der Schnittstellendefinition
  - *bottom-up*:
    - erweitert WSDL um die Möglichkeit semantische Beschreibung anzubringen
    - abstrahiert von der verwendeten Ontologiesprache
- semantische Beschreibung einer Operation durch *IOPE*: *input*, *output*, *precondition*, *effect*

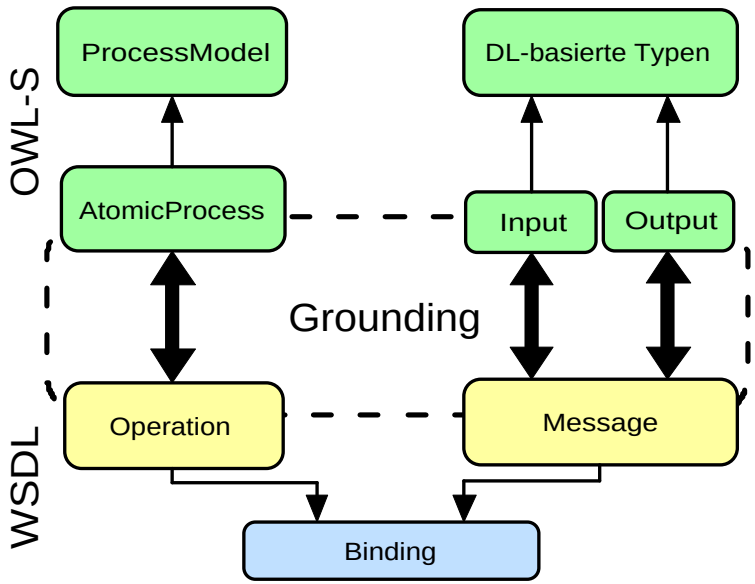
OWL-S – Web Ontology Language for Services (1)

- definiert eine Ontologie zum Beschreiben von Web Services
- Ziel: Finden, Aufrufen und Zusammensetzen von Diensten soll automatisiert möglich sein
- semantische Beschreibung eines Services wird in drei Klassen gegliedert:
  - *Service-Profile*:
    - funktionale Eigenschaften eines Dienstes
      $\Rightarrow$ *IOPE*-Beschreibungen der Operationen
    - nicht-funktionale Eigenschaften und weitere Aspekte, wie z. B. Kategorisierungs-Informationen des Services



OWL-S – Web Ontology Language for Services (2)

- *Service-Model*: beschreibt, wie mit einem Service interagiert werden kann
  - enthält die Prozessbeschreibung: Kontrollstrukturen, atomare Prozesse, etc.
- *Service-Grounding*:
  - entspricht dem Binding-Konzept von WSDL
  - dient der Spezifikation der technischen Details
  - verknüpft die abstrakte semantische Service-BEschreibung mit einer Service-Schnittstelle



WSMO-Ansatz

- besteht aus WSMO, WSML und WSMX
- *WSMO*: definiert eine Ontologie zur semantischen Bescheibung von Web Services
- *WSML*: formale Sprache für die Beschreibung der WSMO-Aspekte
- *WSMX*: die Referenzimplementierung dieses Ansatzes
- *WSMO – Web Service Modeling Ontology*:
  - die vier Hauptelemente:
  - **Ontologien**: dienen zur Spezifikation gemeinsam verwendeter Terminologien
  - $\Rightarrow$ werden von allen Elementen in WSMO für die Beschreibung von Wissen verwendet

WSMO-Ansatz – WSMO

- vier Hauptelemente (forts.):
  - **Web-Service-Beschreibungen**: ermöglichen die Beschreibung funktionaler und nicht-funktionaler Aspekte eines Services sowie seiner Schnittstellen
  - *capability*-Element: beschreibt die Leistung, die ein Service anbietet (basierend auf IOPE)
  - *interface*-Element: beschreibt, wie die Leistung eines Dienstes erbracht wird: $\Rightarrow$ Choreografie und Orchestrierung
  - **Ziele**: ein `goal`-Element beschreibt die Anforderungen eines Clients an einen Service
  - ist ähnlich aufgebaut, wie die Service-Beschreibung (`capability` und `interface`)
  - **Mediatoren**: dienen der Überbrückung von Herterogenität, z. B. bei der Verwendung unterschiedlicher Ontologien

SAWSDL

- bottom-up Ansatz
- erweitert WSDL und XML-Schema um Attribute für die semantische Beschreibung
  - *semantische Annotierung*:
    - *modelReference*-Attribut: dient zu semantischen Annotierung von WSDL- und XML-Schema-Elementen
    - kann in jedem Element verwendet werden – Allerdings ist seine Bedeutung nicht für alle Elemente definiert
    - *Annotierung*: Referenz auf ein Konzept in einer Ontologie
  - *Abbildungen zwischen Datentypen und Ontologien*:
    - *liftingSchemaMapping*: XML-Schema-Element $\Rightarrow$ Ontologie-Konzept
    - *loweringSchemaMapping*: Ontologie-Konzept $\Rightarrow$ XML-Schema-Element

Evaluierung der SWS-Technologien

Kriterien	WSMO	OWL-S	SAWSDL
Vollständige semantische Beschreibung	✓	✓	✓
Keine Beschränkung bei der Wahl der Beschreibungssprache	-	-	✓
basierend auf Standards	-	✓	✓
Standard	-	-	✓
Einfachheit	-	-	✓
Datenmediation	✓	✓	✓
Ontologie-Mediation	✓	✓	✓
Implementierungen	✓	✓	✓
Werkzeuge	✓	✓	✓
Vorgehensweise	top-down	top-down	bottom-up

$\Rightarrow$ SAWSDL ist für die Verwendung im Mittler-Szenario am besten geeignet

METEOR-S

- ein Projekt des LSDIS-Labors der University of Georgia
- Ziel:
  - Entwicklung eines Frameworks für die Konfiguration und Ausführung dynamischer Semantik-basierter Web-Prozesse
  - Einbringung von Semantik in den gesamten WS-Lebenszyklus
- Drei Aspekte werden betrachtet:
  - semantische Annotierung von Services
  - Veröffentlichen und Auffinden semantischer Web Services – Verwendung von Dienstverzeichnissen
  - semantische Erweiterbarkeit von Dienstkompositonen
- Prozessdefinition mit semantischen Service-Templates
- Konfigurationsmodul für die Anpassung der Prozessdefinition durch den Dienstanbieter
- Ausführungsumgebung für dynamische Partnerdienste

METEOR-S – Evaluierung

- ermöglicht die Flexibilisierung einer Dienstkomposition aus der Sicht des Anbieters
- hilft dem Anbieter bei der dynamischen Konfiguration seiner Prozessdefinition
- wesentlicher Unterschied zum Mittler-Szenario:
  - Flexibilisierung wird nicht an den Client weitergereicht
  - Client-Sicht wird nicht betrachtet
- weitere Probleme:
  - Integration in BPEL wird nur auf einer sehr abstrakten Ebene beschrieben
  - konkrete Sprachintegration fehlt
  - keine Implementierung der Konzepte erhältlich – ist aber seit zwei Jahren angekündigt

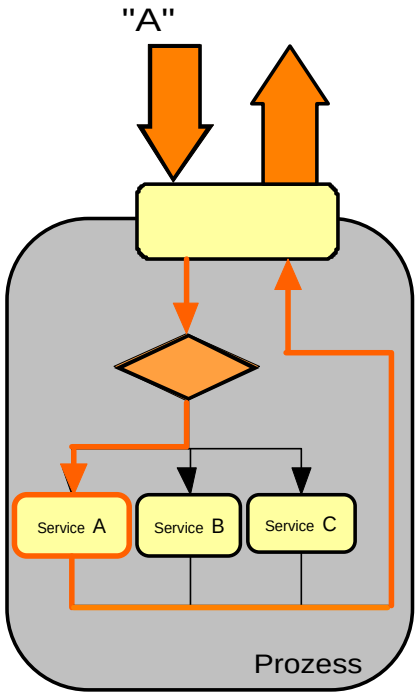
CoSMoS-Projekt

- ist ein Projekt an der University of California, Irvine
- beschäftigt sich mit der dynamischen Komposition von Diensten basierend auf der Dienstanfrage eines Clients
- basiert auf *automated planning*:
  - Komposition ist gänzlich automatisch erstellt
  - Dienstanfrage wird in einen semantischen Graph umgewandelt
  - anhand des semantische Graphen wird versucht eine Komposition zu erstellen, welche die angefragte Funktionalität erfüllt
- ist zu dynamisch für das Mittler-Szenario
- keine Möglichkeit Teildienste der Dienstausführung zu bestimmen

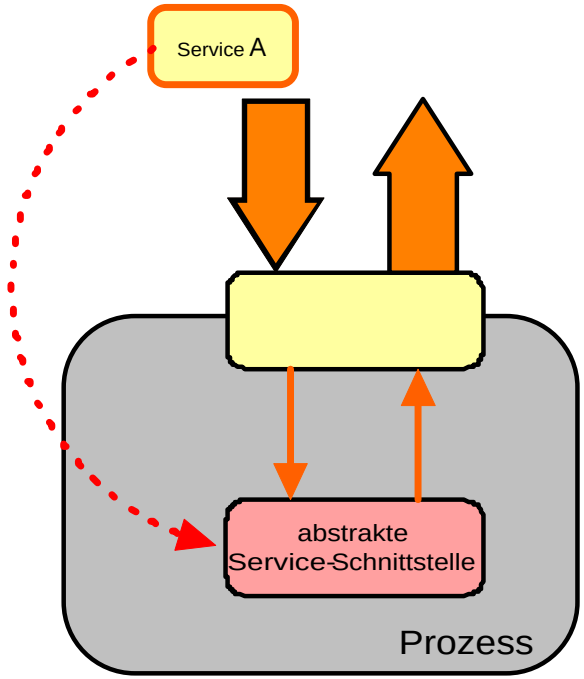
  $\Rightarrow$ Konzepte, die einen *automated planning* Ansatz verfolgen, eigenen sich nicht für das Mittler-Szenario

Motivation – Mittler-Szenario (1)

- Einfluss des Clients auf die Ausführung einer Dienstkomposition: drei mögliche Szenarien:



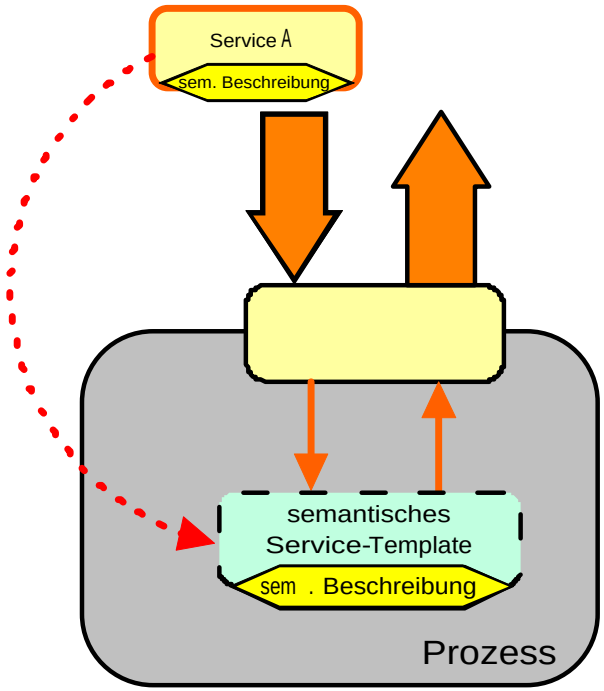
Ausführung des Partner-Services abhängig von dem übergebenen Parameter.



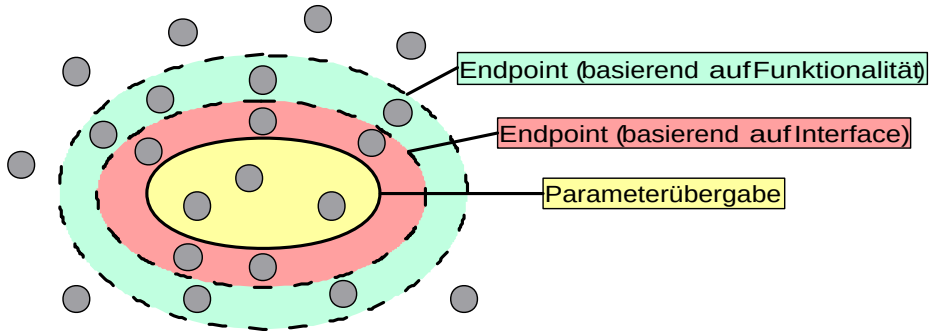
Übergabe eines Endpoints basierend auf der Schnittstellenbeschreibung.

Motivation – Mittler-Szenario (2)

- Einfluss des Clients auf die Ausführung einer Dienstkomposition: drei mögliche Szenarien:



Übergabe eines Endpoints basierend auf der Funktionalität.



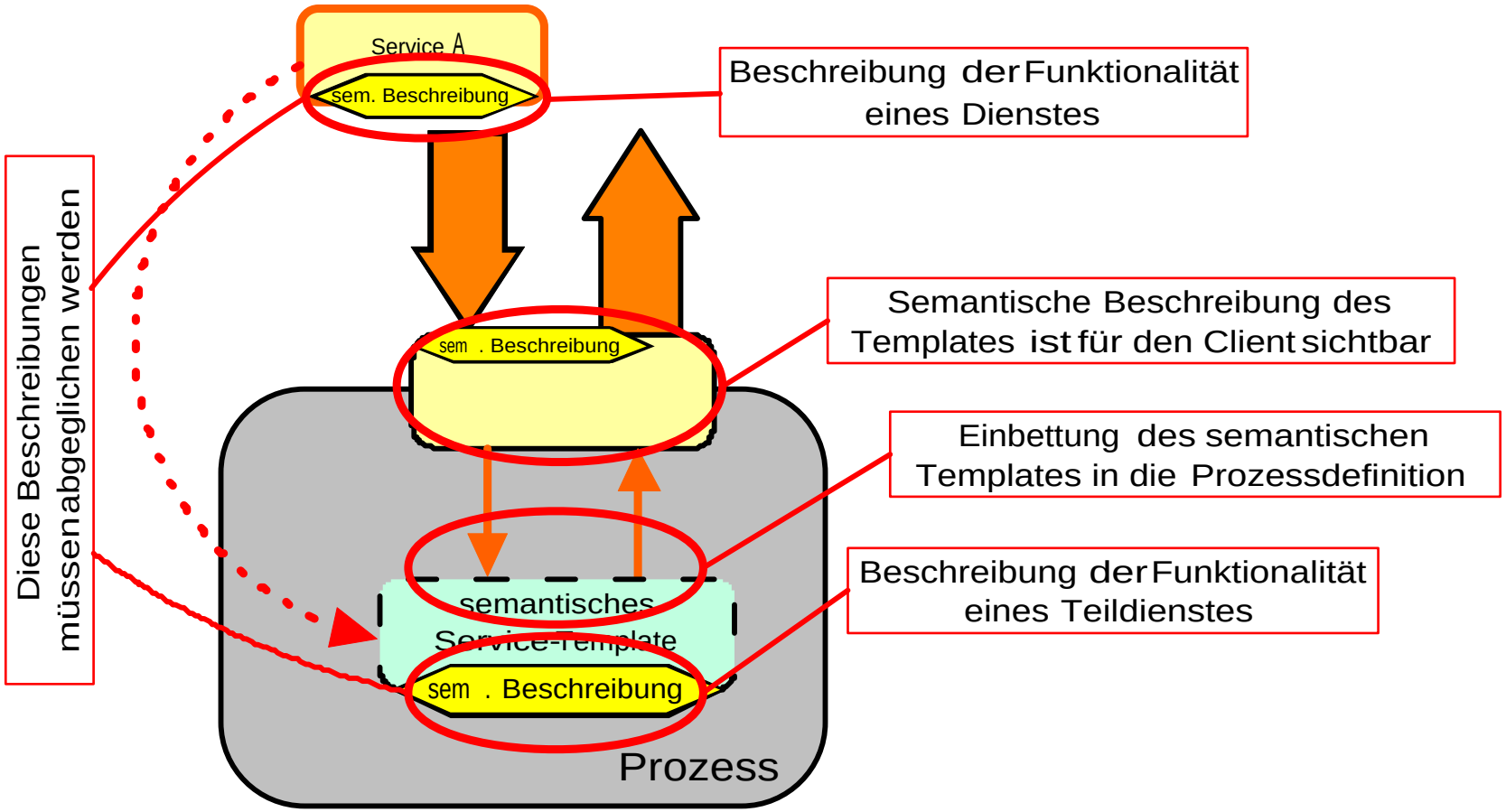
Die Mengen der zur Auswahl stehenden Dienste in den drei Szenarien.

Anforderungen (1)

Anforderungen, die sich aus dem Mittler-Szenario abgeleitet sind

1. die Funktionalität der beteiligten Dienste muss beschrieben werden
2. die Funktionalität eines Teildienstes soll als semantisches Service-Template beschrieben werden
3. Einbettung semantischer Templates in die Prozessdefinition
4. Sichtbarmachung der enthaltenen semantischen Service-Templates
5. Abgleichen der semantischen Beschreibungen der Service-Templates und der übergebenen Dienste

Anforderungen (2)



Umsetzung

zu 1.: Semantische Beschreibung:

- beteiligte Services werden mittels SAWSDL semantisch beschrieben

zu 2. und 3.: Definition eines semantischen Service-Templates:

- Spezifikation des Templates als Spracherweiterung für WS-BPEL
- das `semanticInvokeTemplate`-Element wird in Anlehnung an die `invoke`-Aktivität definiert

Definition eines semantischen Service-Templates (1)

Konstrukte der `invoke`-Aktivität:

- Referenzierung des Partner-Services
  - `partnerLink`-Attribut referenziert den *PortType* des Partners
- Operation, Ein- und Ausgabevariablen:
  - `operation`-Attribut referenziert eine Operation des *PortTypes*
  - Ein- und Ausgabevariablen können WSDL-Nachrichten oder XMLSchema-Elemente sein
- Fehlerbehandlung und Kompensation
  - `catch` und `catchAll` als Elemente innerhalb von `invoke`
  - das `compensationHandler`-Element enthält die Kompensierungsaktion
- Zustandsbehaftete Interaktion
  - mittels Korrelationsmengen, die einfache XML-Schema-Typen als Konversations-Tokens enthalten

Definition eines semantischen Service-Templates (2)

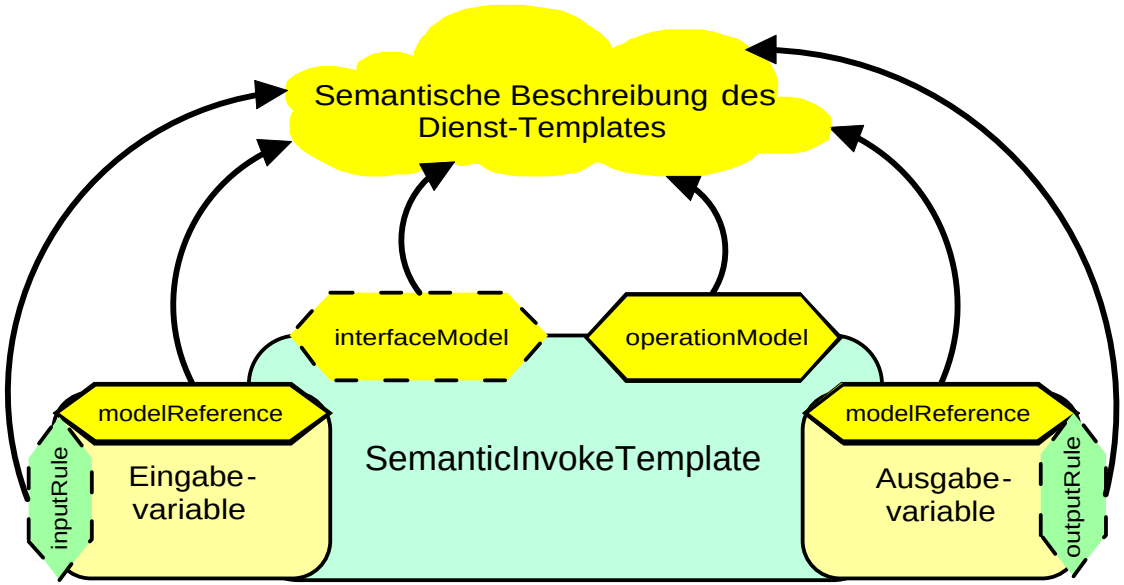
Definition der Erweiterungsaktivität `semanticInvokeTemplate`:

- Referenzierung eines Partner-Services
  - PortType erst zur Laufzeit bekannt ⇒ nicht möglich, fällt weg
- Operation, Ein- und Ausgabevariablen:
  - auch der Name Operation ist erst zur Laufzeit bekannt ⇒ wird ersetzt durch das `operationModel`-Attribut: enthält die semantische Beschreibung der Operation
  - Ein- und Ausgabevariablen: ähnlich wie bei `invoke`. Zusätzlich aber einige Erweiterungen und Einschränkungen.
- Fehlerbehandlung und Kompensation
  - wird von `invoke` übernommen. Nur das `catch`-Element fällt weg.
- Zustandsbehaftete Interaktion
  - lässt sich nicht innerhalb eines semantischen Templates kapseln
  - nicht Bestandteil dieser Arbeit ⇒ fällt weg

Definition eines semantischen Service-Templates (3)

Semantische Beschreibung des Service-Templates:

- interfaceModel, operationModel
- modelReference aus Eingabe- und Ausgabevariable
- input- bzw. outputRule aus Eingabe-, bzw. Ausgabevariable



Definition eines semantischen Service-Templates (4)

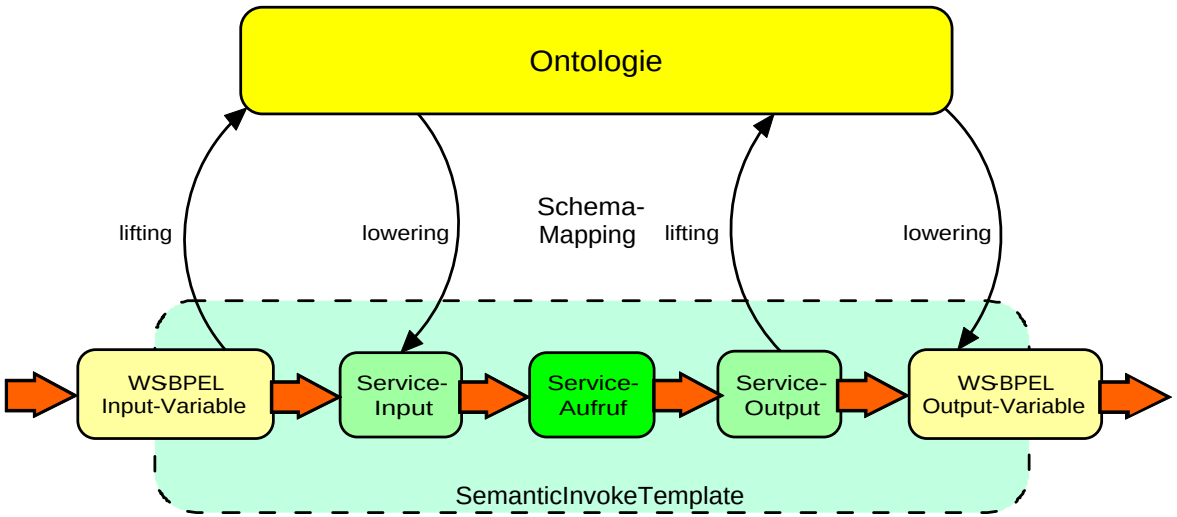
Semantische Beschreibung des Service-Templates (forts.):

- Annotierung der Elemente wie bei SAWSDL
  $\Rightarrow$ Referenz auf Konzepte in einer Ontologie
- inputRule und outputRule referenzieren SWRL-Regeln
- input- bzw. outputRule aus Eingabe-, bzw. Ausgabevariable

Definition eines semantischen Service-Templates (5)

Abbildungen zwischen XML-Schema-Datentypen und Ontologien:

- Datenmediation muss gewährleistet sein



Definition eines semantischen Service-Templates (6)

Abbildungen zwischen XML-Schema-Datentypen und Ontologien (forts.):

- SAWSDL ermöglicht die Angabe von Schema-Mappings mittels der Attribute `liftingSchemaMapping` und `loweringSchemaMapping`
- das *Lifting* ist eine Abbildung von Datentypen auf Ontologien
- das *Lowering* bildet Ontologien auf Datentypen ab
- für beide Abbildungsarten kann XSLT verwendet werden
  - falls auch in der Ontologie XML als Darstellungsform verwendet wird

Veröffentlichung der enthaltenen Service-Templates (1)

Clients sollen erfahren, welche semantischen Templates in einem Mittler-Service enthalten sind

- Veröffentlichung über die WSDL-Schnittstelle des Mittler-Service:
  - Erweiterung der Schnittstelle: Definition des neuen Elements *SemanticEndpoint*
  - für jedes enthaltene Service-Template wird ein *SemanticEndpoint* in der Schnittstelle veröffentlicht
  - ein *SemanticEndpoint* besteht aus einer Endpoint-Referenz einem WSDL Dokument

Veröffentlichung der enthaltenen Service-Templates (2)

- Veröffentlichung über die WSDL-Schnittstelle des Mittler-Service (forts.):
  - ein *SemanticEndpoint* enthält die semantische Beschreibung eines Templates
    $\Rightarrow$ Client hat Zugriff auf die Beschreibung der enthaltenen Templates
  - beim Aufruf: Übergabe einer Endpoint-Referenz und seiner semantisch annotierten WSDL-Beschreibung
    - WSDL Übergabe als Referenz oder eigebettet

Abgleich der semantischen Beschreibungen

Grundregeln:

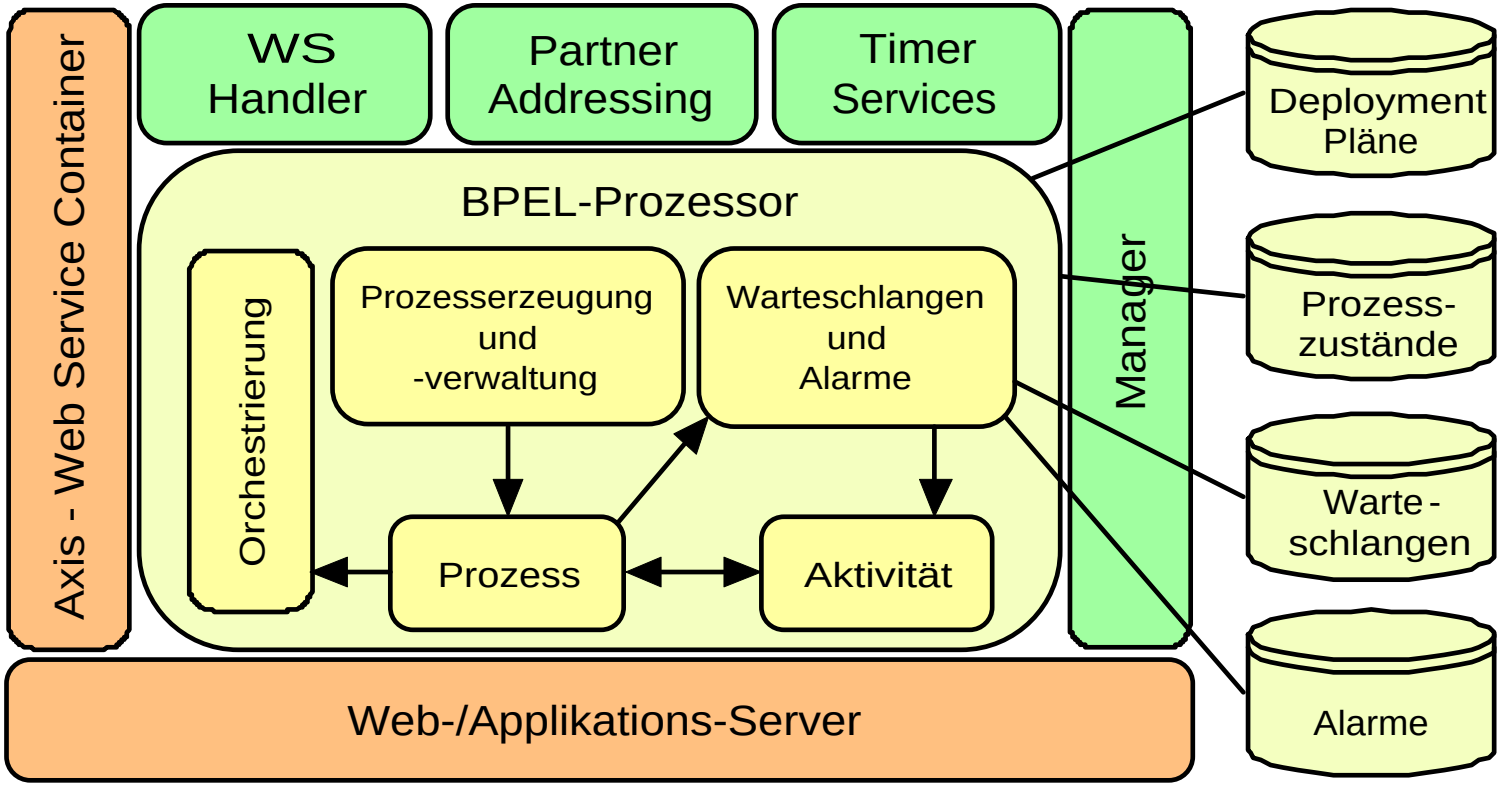
- alle semantischen Annotierungen des Templates müssen berücksichtigt werden
- Annotierungen im WSDL-Dokument eines übergebenen Dienstes werden nicht berücksichtigt, sofern diese nicht im Template angegeben sind
- Überprüfung: Äquivalenz ist gegeben, wenn
  - Template-Konzept äquivalent zu Dienst-Konzept oder
  - Dienstkonzept ein Subkonzept der Template-Konzepts ist

BPEL-Engine und Werkzeuge

Verwendete BPEL-Engine und Werkzeuge

- für die Implementierung der Erweiterung wird ActiveBPEL 4.0 verwendet
- Werkzeuge:
  - **SAWSDL4J**: Java-Implementierung des SAWSDL-Standards
  - **Jena-Framework**: bietet eine OWL-API, die für die Verarbeitung von OWL-Ontologien benötigt wird
  - **Pellet**: OWL-Reasoner
  - **Apache Axis**: SOAP-Engine – wird auch von ActiveBPEL verwendet
  - **Apache Tomcat**: Ausführungsumgebung für ActiveBPEL

Architektur der ActiveBPEL-Engine



Funktionsweise der ActiveBPEL-Engine (1)

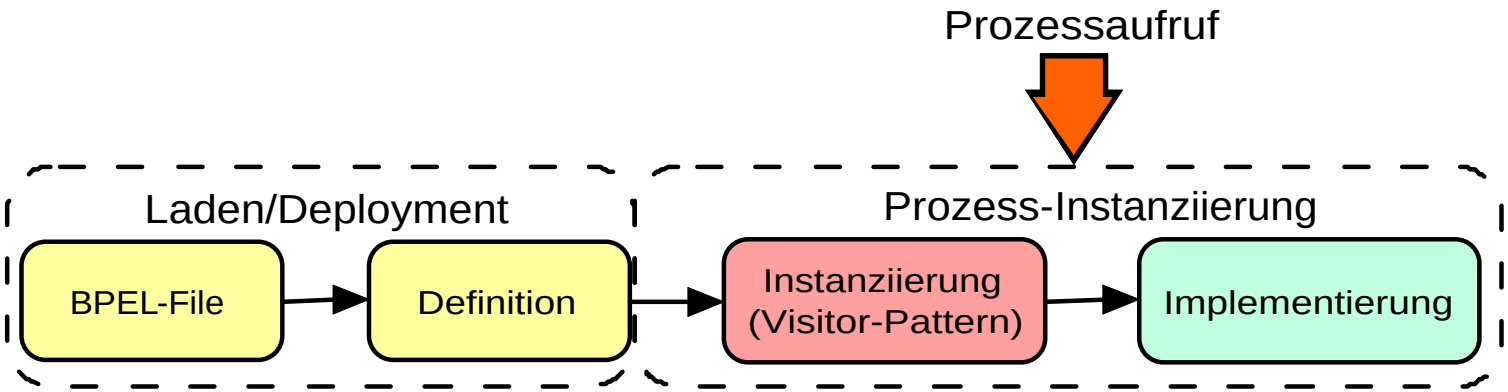
Deployment einer Prozessdefinition:

- Erstellen eines Geschäftsprozess-Archivs *(*.bpr)*:
  - BPEL-Prozessdefinition *(*.bpel)*
  - *Process Deployment Description (*.pdd)*
  - Katalog der benötigten Ressourcen *(catalog.xml)*
- Geschäftsprozess-Archiv muss in das Verzeichnis *$TOMCAT_HOME/bpr* kopiert werden

Funktionsweise der ActiveBPEL-Engine (2)

Erzeugung einer Prozessinstanz:

- Erstellen eines Geschäftsprozess-Archivs *(*.bpr)*:
  - BPEL-Datei wird eingelesen
  - Objektrepräsentation der Prozessdefinition wird erstellt
  - beim Prozessaufruf wird mittels der Prozessdefinition eine Prozessinstanz erzeugt
- Geschäftsprozess-Archiv muss in das Verzeichnis *$TOMCAT_HOME/bpr* kopiert werden



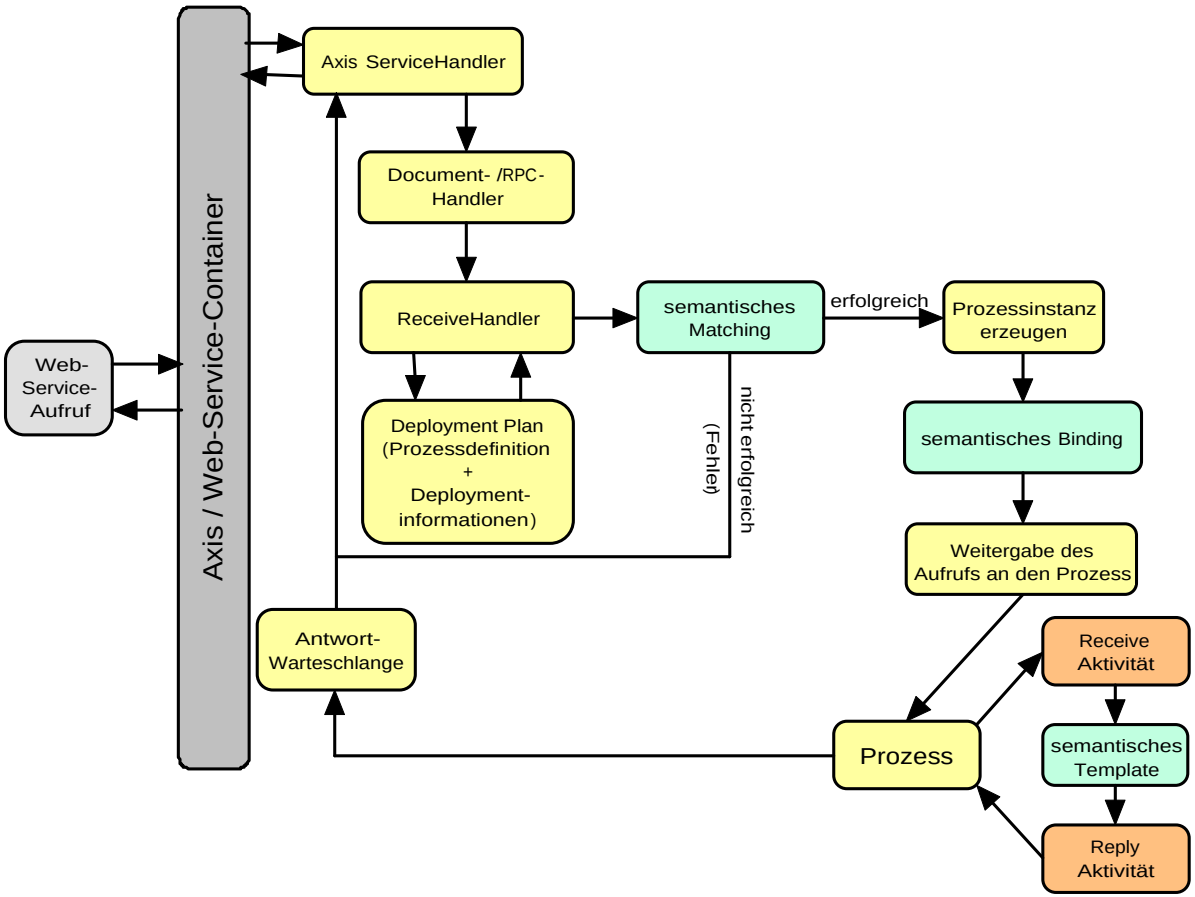
Implementierung des *semanticInvokeTemplate*-Elements

- **Def-* und **Impl-Klasse*:
  - **Def-Klasse*: implementiert die Erweiterungsaktivität als Definitionselement
  - **Impl-Klasse*: implementiert das Laufzeitverhalten in einer Prozessinstanz
- `SemanticInvokeTemplateImpl`:
  - erbt von `AeActivityInvokeImpl` und überschreibt die Methoden *execute()*, `onMessage()` und `getInputMessageData()`
  - enthält die Informationen eines *SemanticEndpoints*: Endpoint-Referenz und semantisches WSDL-Dokument
  - wird erst zur Laufzeit an eine Dienstimplementierung gebunden

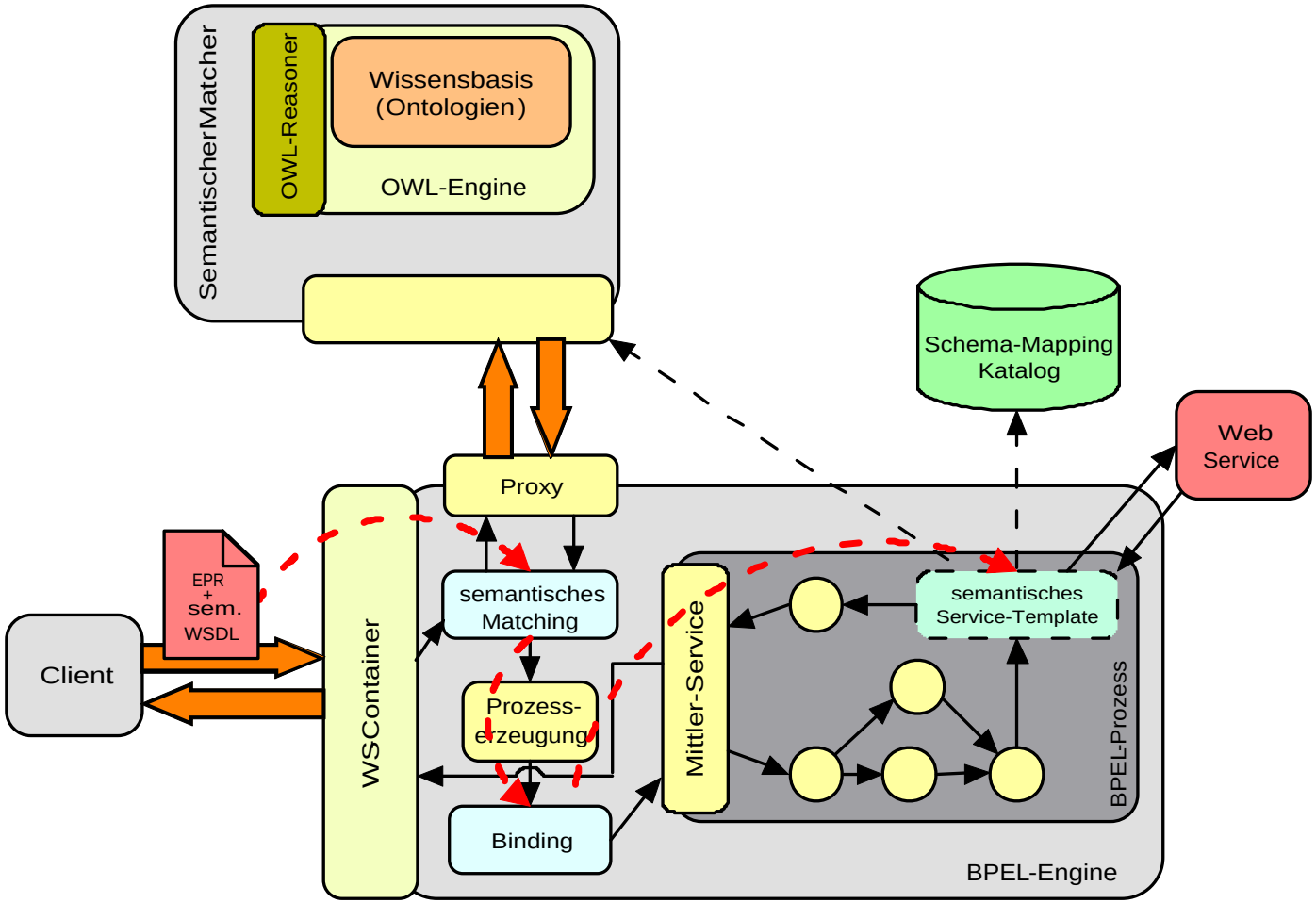
Weitere Aspekte

- semantisches Matching und Binding:
  - semantisches Matching ist als eigenständige Komponente außerhalb der BPEL-Engine implementiert
  - semantische Informationen werden aber in der BPEL-Engine extrahiert und an die Matching-Komponente übergeben
  - das Binding der *SemanticEndpoints* an die Service-Templates erfolgt nach der Erzeugung einer Prozessinstanz
- Prozess-Katalog:
  - Ressourcen, die nur im Kontext einer Prozessinstanz gelten
  - ist an eine Prozessinstanz gebunden
- Schema-Mapping-Katalog
- eigener *InvokeHandler*

Ablaufschema der semantisch erweiterten ActiveBPEL-Engine



Architektur des Mittler-Szenarios



Ausblick

- Ontologie-Entwicklung
- Einbringung von Ontologie-Mediation
- Umgang mit Verträgen zwischen dem Client und dem übergebenen Dienst
- Umgang mit Zugangsdaten – Sicherheit und Vertraulichkeit
- Erweiterung der Matching-Kriterien
  - funktionale und nicht-funktionalie Kriterien, die ein Client beim Aufruf übergibt
- GUI-Unterstützung

Fragen?