



Dienstorientierte Integration: Erweiterte Web-Service-Konzepte

Andreas Schmidt
WS 2011/12

- UDDI
- BPEL4WS
- Semantische Dienstbeschreibungen
- Fazit zu Web Services



UDDI
Universal Description,
Discovery and Integration

UDDI: Überblick

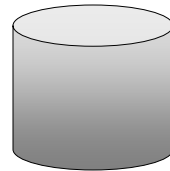
- **U**niverselle, zentrale "E-Business"-Suchmaschine mit verteilter Konzeption (UDDI Business Registry)
 - Informationen über Unternehmen abrufen (**D**escription)
 - Geschäftspartner auffinden (**D**iscovery)
 - Geschäftsprozesse über Unternehmensgrenzen hinaus beschreiben und nutzbar machen (**I**ntegration)
- Entwicklung begann im Frühjahr 2000
 - Ariba, IBM, Microsoft
 - Erste Registry im Mai 2001

UDDI: Überblick (2)

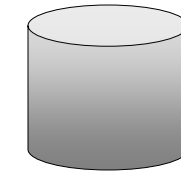
- Universeller Verzeichnisdienst
 - *white pages*: Informationen über Unternehmen (Namensregister, Details, Kontaktinformationen)
 - *yellow pages*: »Branchenbuch« - Ordnung der Unternehmen nach Technologiecodes, Branchentaxonomien oder geographischem Ort
 - *green pages*: Geschäftsmodell, technische Informationen über Nachrichtenformate, Protokolle (=> WSDL), Informationen über Geschäftsprozesse
 - *service type registrations*: Referenzen auf Beschreibungen des jeweiligen Typs
- primärer Zweck ist nicht unbedingt das *automatische* Auffinden

- Informationen über das anbietende Unternehmen
 - auch Taxonomie nach Branche o.ä. möglich
- Beschreibung des angebotenen Dienstes
 - nicht nur für Web Services
 - Kategorisierung des Dienstes (Thesauri) nach
 - Branche
 - Art der Dienstleistung
 - geographischer Ort
 - Technische Schnittstelle (z.B. WSDL, aber auch natürlich sprachliche Beschreibungen)
- Protokoll zum Veröffentlichen und Nachschlagen von Dienstinformationen (Basis: SOAP)
 - verteilte Registries

Nutzung der Daten über Suchmaschinen oder aus Anwendungen direkt heraus, um Dienste anderer Unternehmen zu finden und in eigene Anwendungen zu integrieren



Business Registrations

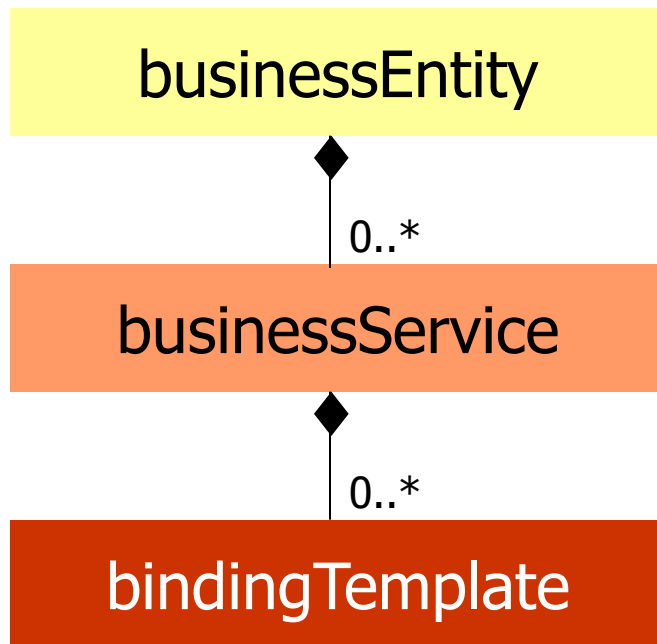


Service Type Registrations

Unternehmen registrieren sich und beschreiben sich und die Dienste, die sie anbieten

Standardisierungsgremien definieren Diensttypen (tModels)

Datenstrukturen



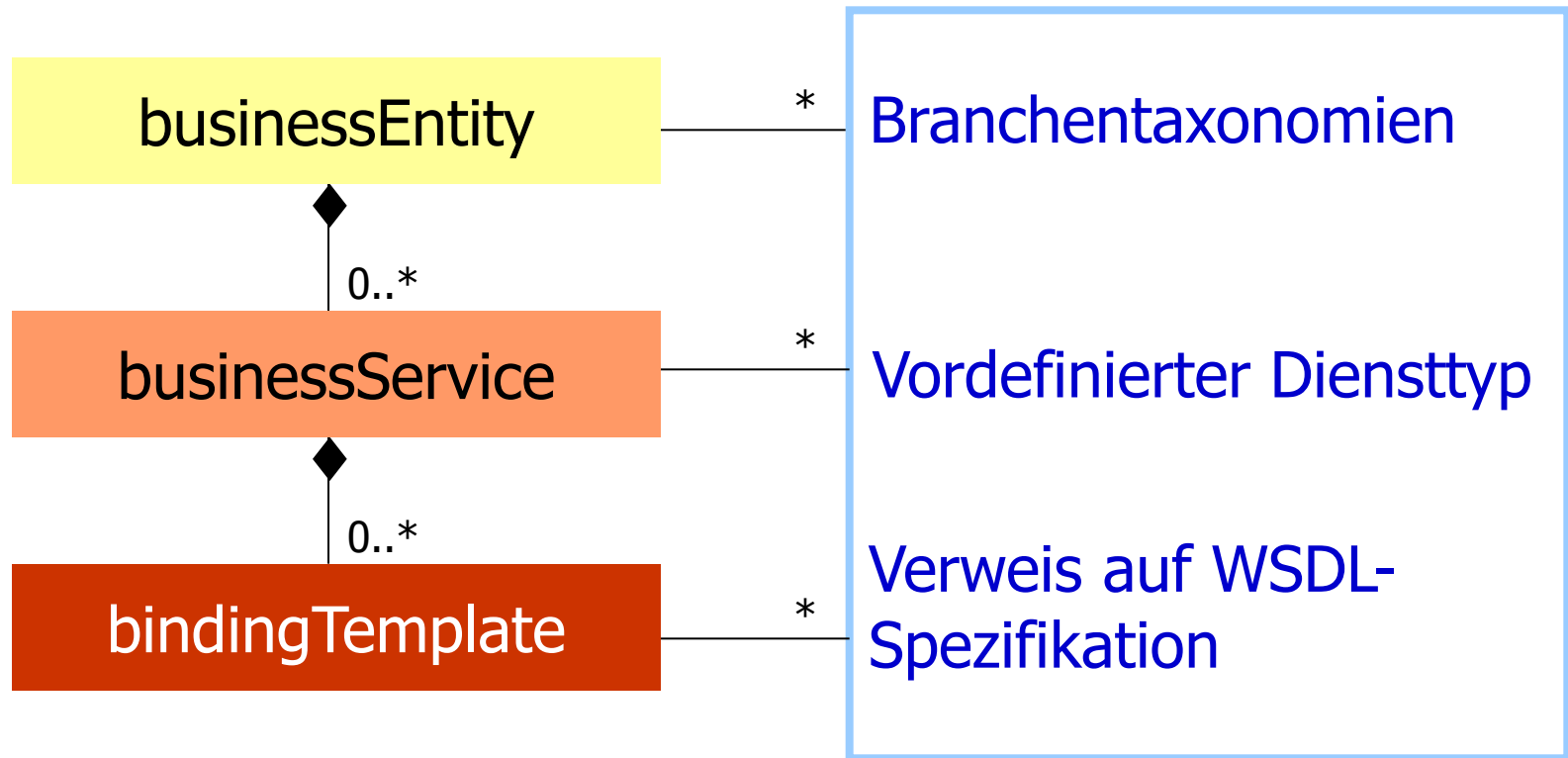
Name, Beschreibung, Kontakt
URL-Verweise auf andere businessEntities
Zuordnung zu Taxonomien

Name, Beschreibung,
Zuordnung der Dienste zu Kategorien

Name, Beschreibung,
Kommunikationsmöglichkeiten (http, phone)
Zugriffspunktdefinition (Adresse)

Technology Models (tModel)

- tModels sind zunächst nur eindeutige Identifikatoren



Musterablauf Protokoll

Browse
Suchen

find_business
find_service

Drill Down
Details abrufen

get_businessDetail
|
get_serviceDetail

Invocation
Dienst aufrufen

get_bindingDetail
liefert WSDL-Information
anschließend SOAP

Zwischenfazit



Was ist das Neue daran?



- dienstorientierte Architektur
 - Web Services werden als echte Dienstleistungen begriffen, die man nach außen auch anbieten kann
- Granularität
 - grobe Granularität
- Rahmenbedingungen
 - Web-Umgebungen
 - Heterogenität
 - Firewall-Problematik
- (Die Technik ist noch relativ einfach)

Vergleichbare Technologien



■ CORBA

- weniger Autonomie der Dienste
- für lose Kopplung wenig geeignet
- performanter, dadurch feingranularere Verteilung
- trotz IIOP keine wirkliche Internet-Technologie

■ COM/DCOM

- ebenso eher Intranet-Technologie
- keine Plattformunabhängigkeit

- Web Services (insbesondere SOAP) treten auch mit dem Argument an, daß die Firewall-Problematik, die den Einsatz anderer Middleware-Technologien wie CORBA schwierig machte, umgangen wird
 - HTTP als Transportprotokoll ist meist freigeschaltet
 - aber: HTTP büßt immer mehr seine Harmlosigkeit ein
 - Dienste mit beliebigen Auswirkungen können aufgerufen werden
 - Sinn einer Firewall?
- => Damit wird wohl auch SOAP zukünftig Firewall-Restriktionen in Kauf nehmen müssen.

Vorteile von Web Services

- breite Unterstützung der Software-Hersteller
 - Microsoft, IBM, Oracle, ...
- niedrige Einstiegshürden
 - SOAP leicht zu verstehen, wenn man XML kennt
 - WSDL und UDDI sind zunächst einmal optional, um Web Services zu nutzen!
 - leicht kombinierbar mit existierenden Technologien (Servlets, JSP/ASP)
- durchgängige Verwendung von XML
 - und die meisten Anwendungen können in der einen oder anderen Form bereits XML »sprechen«
- für Internet-Umgebungen entwickelt
 - Standard-Protokolle
 - Rahmenbedingungen berücksichtigt

- Sicherheit?
 - mit HTTPS (SSL) läßt sich Client- und Serverauthentisierung und verschlüsselter Datentransfer auf Transport-Protokollebene erreichen
 - XML Encryption / Digital Signature kann auf Anwendungsebene eingesetzt werden
- Performanz?
 - Web Services erlauben nur grobgranulare Verteilung
 - Kompressionstechniken für große Datenmengen
- Erweiterte Konzepte zur Verteilung?
 - Objektreferenzen?
- Wildwuchs an neuen Standards

Die Web-Service-Welt

Geschäftsprozesse

BPEL
WS-Choreography
WS-Choreography Description Language

Management

BPML
WSMF
WS-Management
MUWS
WS-Events
MOWS
WSMF-WSM

Sicherheit

WS-Security
WS-Trust
WS-Security Policy
WS-Federation
WS-Secure Conversat.

Zuverlässigkeit

WS-Reliability
WS-Reliable Messag.

Transaktionen

WS-Business Activity
WS-Atomic Transact.
WS-Coordination

Ressourcen

Web Services
Resources Framework

Metadaten

WSDL
WS-Discovery
UDDI
WS-Policy
WS-Metadata-Exchange

Nachrichten

SOAP
WS-Notification
WS-Adressing

Dienstintegration



- Idee einer Programmierung auf höhere Ebene
- Kombination von Dienstaufrufen zur Erbringung eines »höherwertigen« Dienstes
- Stichwort **Orchestrierung**:
 - neuer komplexer Dienst entsteht auf der Basis von Aufrufen von Einzeldiensten
 - Ausführung wird von einem Interpreter („dem Dirigenten“) gesteuert
- Vorteil:
 - einfache Anpassungen im Rahmen des Betriebs können mittels graphischen Werkzeugen erledigt werden
 - langfristig: bessere Verwaltung von Abhängigkeiten möglich

- In komplexen Szenarien, in denen autonome Dienste zusammenwirken, braucht es Regeln für die Interaktion
 - Analogie: soziale Systeme brauchen auch Konventionen
- Stichwort **Choreographie**:
 - beschreibt das Zusammenwirken von Diensten von außen
 - betrachtet nur das beobachtbare Verhalten von Diensten
 - „Choreographie ist etwas, an das sich alle Beteiligten halten“
 - im Gegensatz zu einem Dirigenten, der etwas vorschreibt
- Beschreibt also Protokolle!

BPEL4WS
Business Process Execution Language
for Web Services



- BPEL4WS kümmert sich (im wesentlichen) um die Orchestrierung von Web Services
- Grundperspektive von BPEL4WS
 - Wie wirken unterschiedliche Web Services im Rahmen von Geschäftsprozessen zusammen?
 - Prozeßbeschreibungssprache (abstrakt, aber auch ausführbar)

Grundschemata

```
<process>
```

```
<partners>
```

```
...
```

```
</partners>
```

```
<containers>
```

```
...
```

```
</containers>
```

```
<faultHandlers>
```

```
...
```

```
</faultHandlers>
```

```
<flow>
```

```
...
```

```
</flow>
```

```
</process>
```

Deklariert die beteiligten Partner und ihrer **Rollen** mittels in WSDL deklarierten ServiceLinkTypes

```
<partner name="" serviceLinkType=""  
myRole="" partnerRole="">
```

Zustandsvariablen der Prozeßinstanz

```
<container name="" messageType="" />
```

Fehlerbehandlung: Definiert, welche Nachricht im Fehlerfall zurückgegeben werden

Aktivitäten

Primitive für Aktivitäten: Dienstaufrufe

■ Parameter

- partner,
- portType
- operation
- inputContainer, outputContainer

<receive ... >

- Wartet auf eine eingehende Nachricht
- Initiiert Aktivitäten
- Alternative: <pick> <onMessage ...> </onMessage > </pick>
 - Wartet auf eine von mehreren möglichen Nachrichten

<reply ... >

- Schickt eine Antwort an den entsprechenden Dienst

<invoke ... >

- Ruft einen Dienst auf

Primitive für Aktivitäten: Programmiersprachenelemente

- **<assign>**
 - Zuweisungsoperation <from> ... <to>
 - Flexible Möglichkeiten über XML-basierte Strukturen

```
<from container="" part="ncname" query="queryString" />
<from expression="" />
<to container="" part="" query="" />
```
 - XPath als Ausdruckssprache und Anfragesprache
- **<switch>...<case condition=""> ... </switch>**
 - Auswahlkonstrukt
- **<while condition="">**
 - Schleife

- Fehlerbehandlung
 - Analog zu Javas Exception-Mechanismus

```
<throw faultName="" />
```

```
<catch faultName="">...</catch>
```

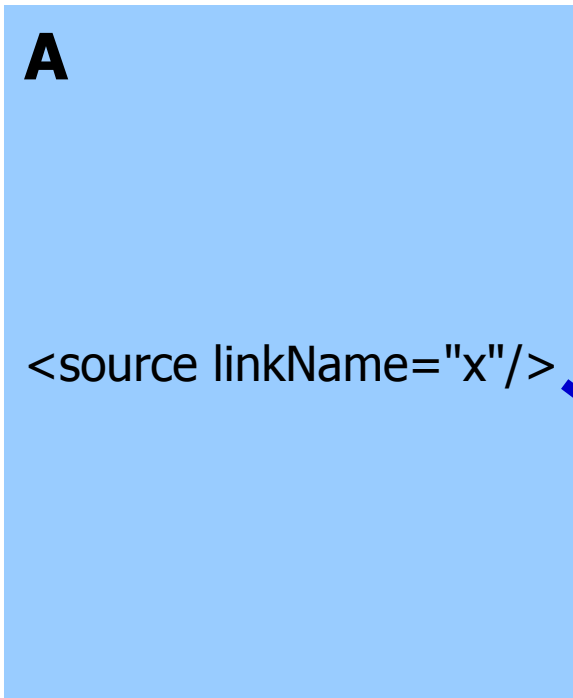
```
<catchAll>...</catchAll>
```

- BPEL4WS erlaubt die Spezifikation nebenläufiger Prozesse (`<flow>`)
- Sequentielle Prozeßabschnitte werden in `<sequence>`-Elemente eingeschlossen
- Will man Barrieren, so gibt es hierfür die Möglichkeit, Abhängigkeiten zu definieren
 - Am Beginn eines Prozeßabschnittes: Verbindungsdeklarationen mittels `<link>`
 - Verknüpfungspunkte: `<source>` bzw. `<target>`
 - Entspricht in Java `notify()` bzw. `wait()`
 - Zusätzliche Bedingungen sind auch möglich

Synchronisation (2)

<flow>

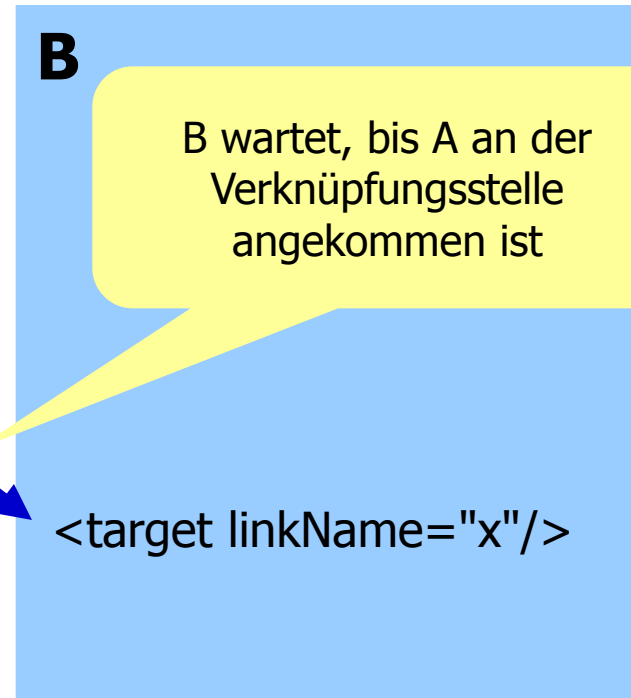
<sequence>



</sequence>

</flow>

<sequence>



</sequence>

- BPEL4WS unterstützt Transaktionen
- Allerdings: keine ACID-Transaktionen
 - Atomizität ist für langlebige Transaktionen problematisch
 - Keine Transaktionsunterstützung bei den Diensten voraussetzbar
 - z.B. Zwei-Phasen-Commit setzt ACID-fähige Dienste voraus!
- Stattdessen:
 - Offen geschachtelte Transaktionen ("Sagas")
 - Vgl. Vorlesung Transaktionsverwaltung
 - Zwischenzustände werden bereits sichtbar
 - Im Rücksetzfall werden Kompensationen eingesetzt, die einen konsistenten Zustand wiederherstellen (sollen)

Transaktionen (2)

```
<scope name="A">
```

```
  <compensationHandler>
```

Aktionen, die im Fehlerfall aufrufen sind

```
</compensationHandler>
```

Aktivitäten

```
  <scope name="B">
```

```
    <compensationHandler>...</compensationHandler>
```

Aktivitäten

```
    <compensate/>
```

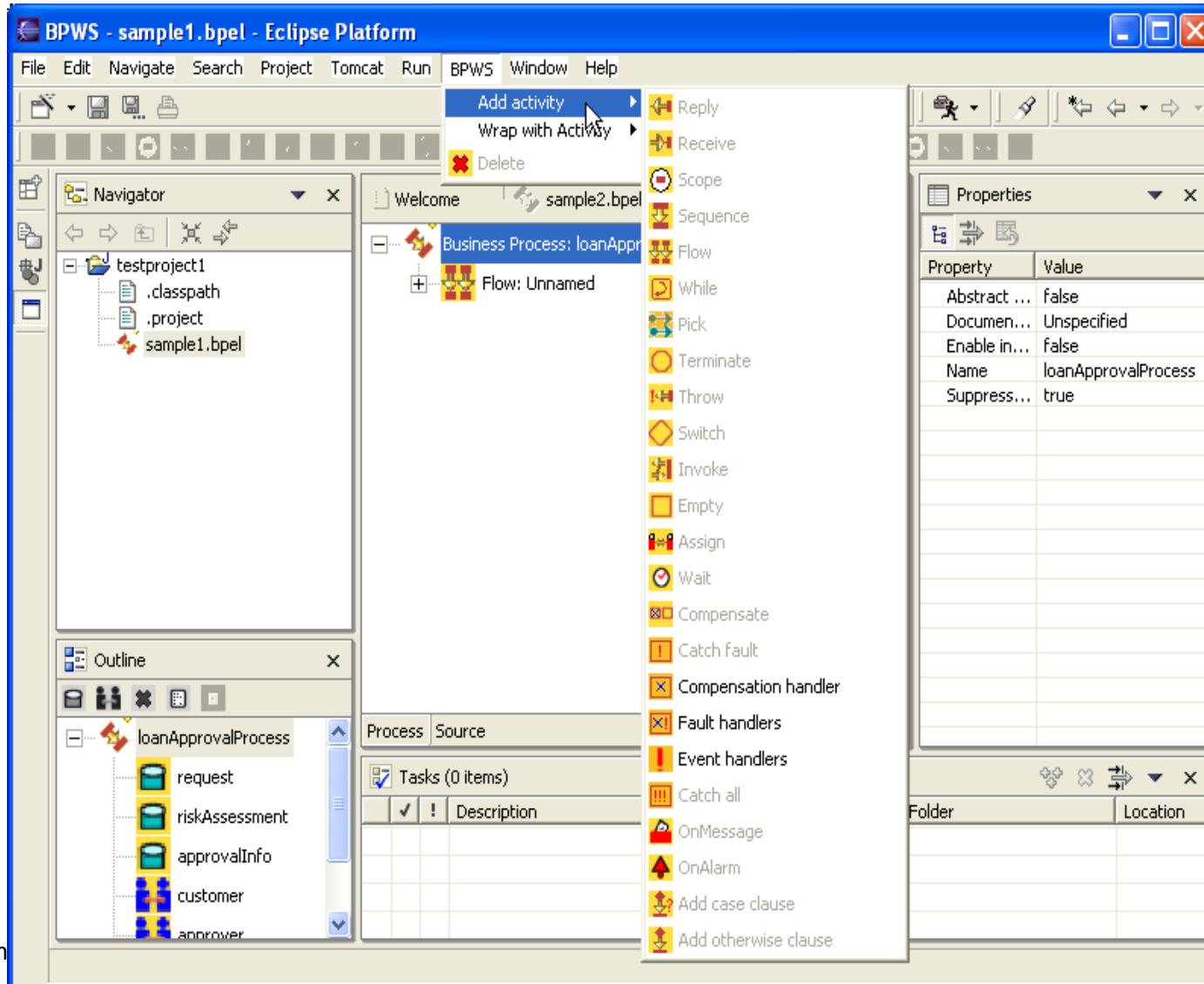
```
  </scope>
```

Aktivitäten

```
</scope>
```

- "Schlüssel" für Prozeßinstanzen
 - Spezifikation, welche Attribute (in den Nachrichtenparametern) es erlauben, die empfangene Nachricht einer Prozeßinstanz zuzuordnen
 - Beispiel: Bestellnummer für Bestellung, Statusanfrage etc.
 - Bei Empfang wird das entsprechende Attribute extrahiert und nach vorhandenen Prozeßinstanzen gesucht
 - Dadurch kann eine mehrschrittige verbindungslose Kommunikation stattfinden
- Korrelationen werden den <receive>, <reply> und <invoke>-Elementen zugeordnet

Grafische Modellierung



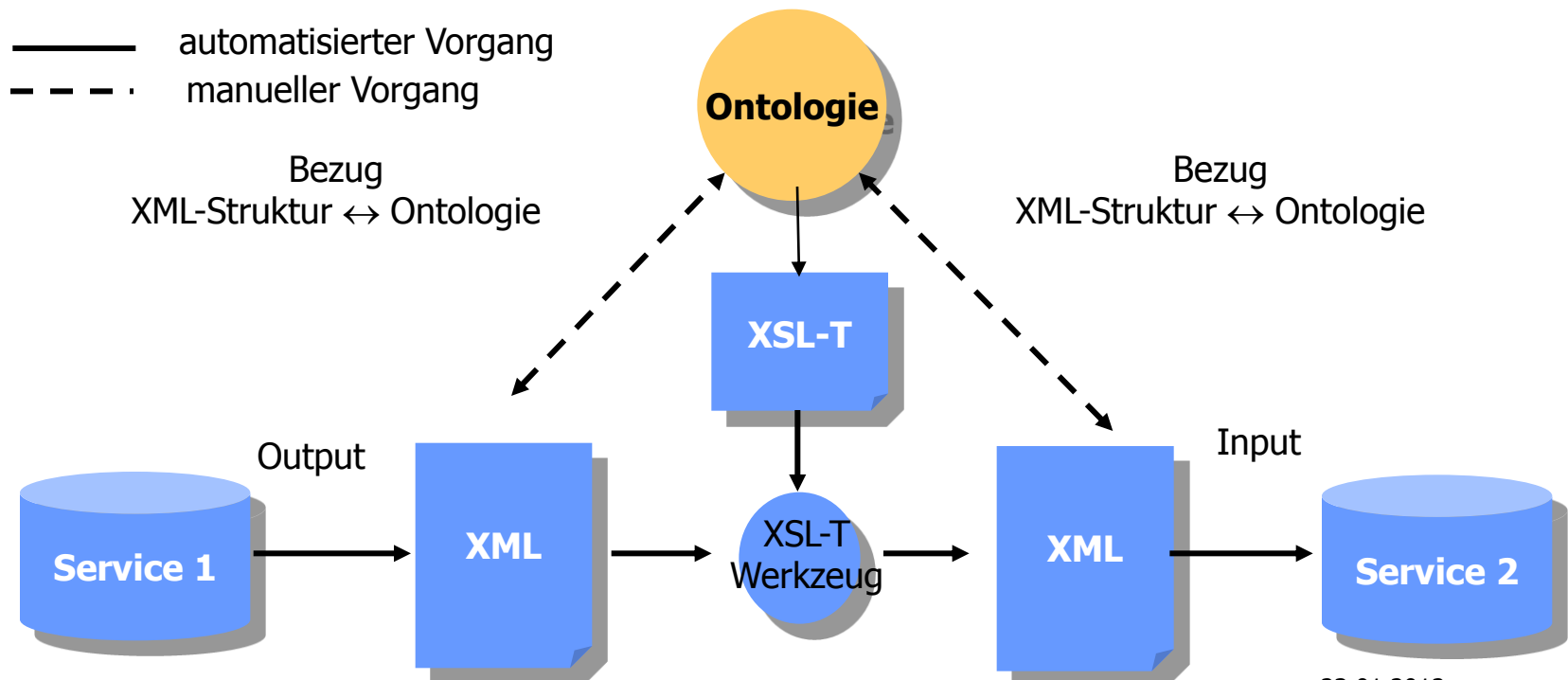
Dienstorientierte Architekturen und Semantik



- Bislang hat sich die Beschreibung von Web Services auf syntaktische Aspekte beschränkt
 - Überprüfung, ob syntaktisch korrekter Aufruf
 - Generierung von sprachspezifischen Bindings
- Aber:
 - keine Suche nach Diensten, die etwas Bestimmtes tun
 - keine automatisierte Kopplung
 - keine Prüfung von Qualitätsbeschreibungen
- Idee: Erweiterung der Beschreibungskonzepte für Web Services durch ontologiebasierte Verfahren, so dass eine maschinenverarbeitbare semantische Beschreibung entsteht.

Was wollen wir semantisch beschreiben?

- **1. Semantik der übertragenen Parameter (Informationsebene)**
 - Mittel: semantische Annotation der Nachrichtenschemata
 - Zweck: Interoperabilität/Mediation



Was wollen wir semantisch beschreiben?

- **2. Semantik der Dienstfunktionalität (Anwendungsebene)**
 - Mittel: Annotation mit Vor-/Nachbedingungen
 - Zweck: Dienstfindung und –komposition

- **Bestelldienst**
 - Vorbedingung: Buch im Katalog
 - Nachbedingung: Buch bestellt

- **Lieferdienst**
 - Vorbedingung: Buch bestellt, Buch im Lager
 - Nachbedingung: Buch bei Kunde

- **Kopplung: Buchkauf**

Was wollen wir semantisch beschreiben?



- 3. Semantik der Dienstgüte
 - Mittel: formale Beschreibung der Dienstgüte / SLAs
 - Zweck: Dienstfindung und Überprüfbarkeit

- Zusätzliche Angaben:
 - garantierte (technische) Antwortzeit
 - garantierte Lieferzeit

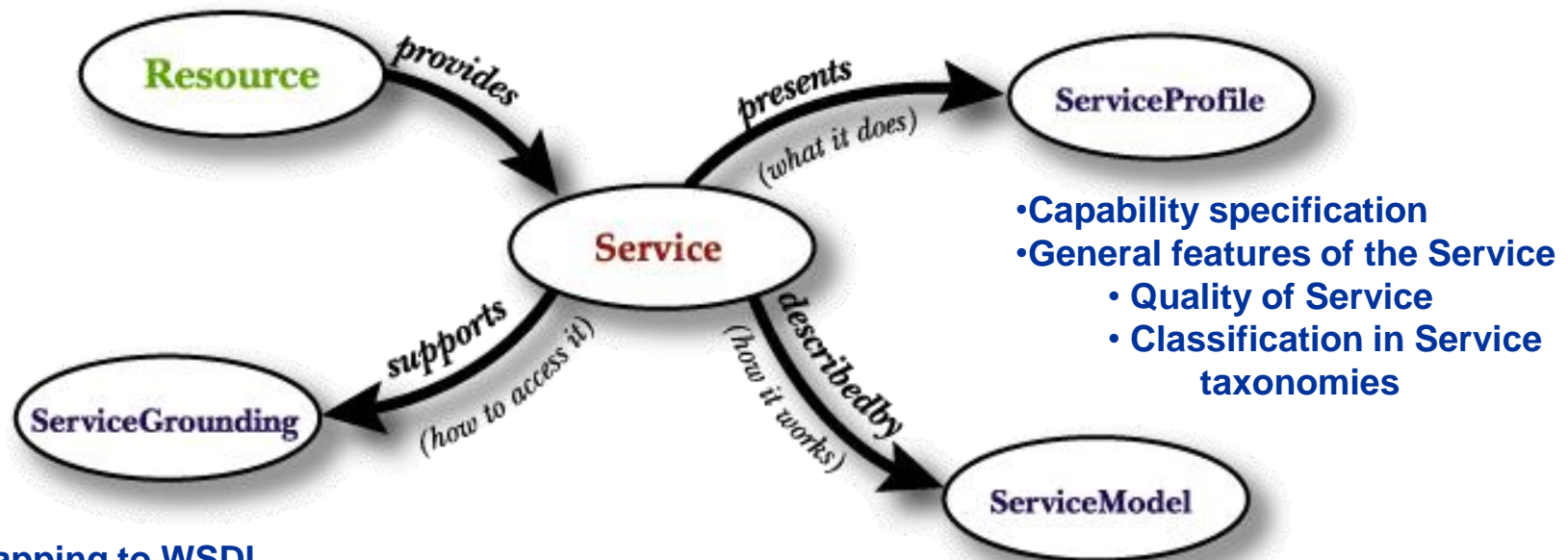
Was wollen wir semantisch beschreiben?



- **4. Ausführungssemantik**
 - Mittel: formale Modellierung der Abläufe
 - Zweck: Validierung, Verifikation

- Einsatz von Techniken aus der Programmverifikation
 - sehr aufwendig

Beispiel: OWL-S



- Mapping to WSDL

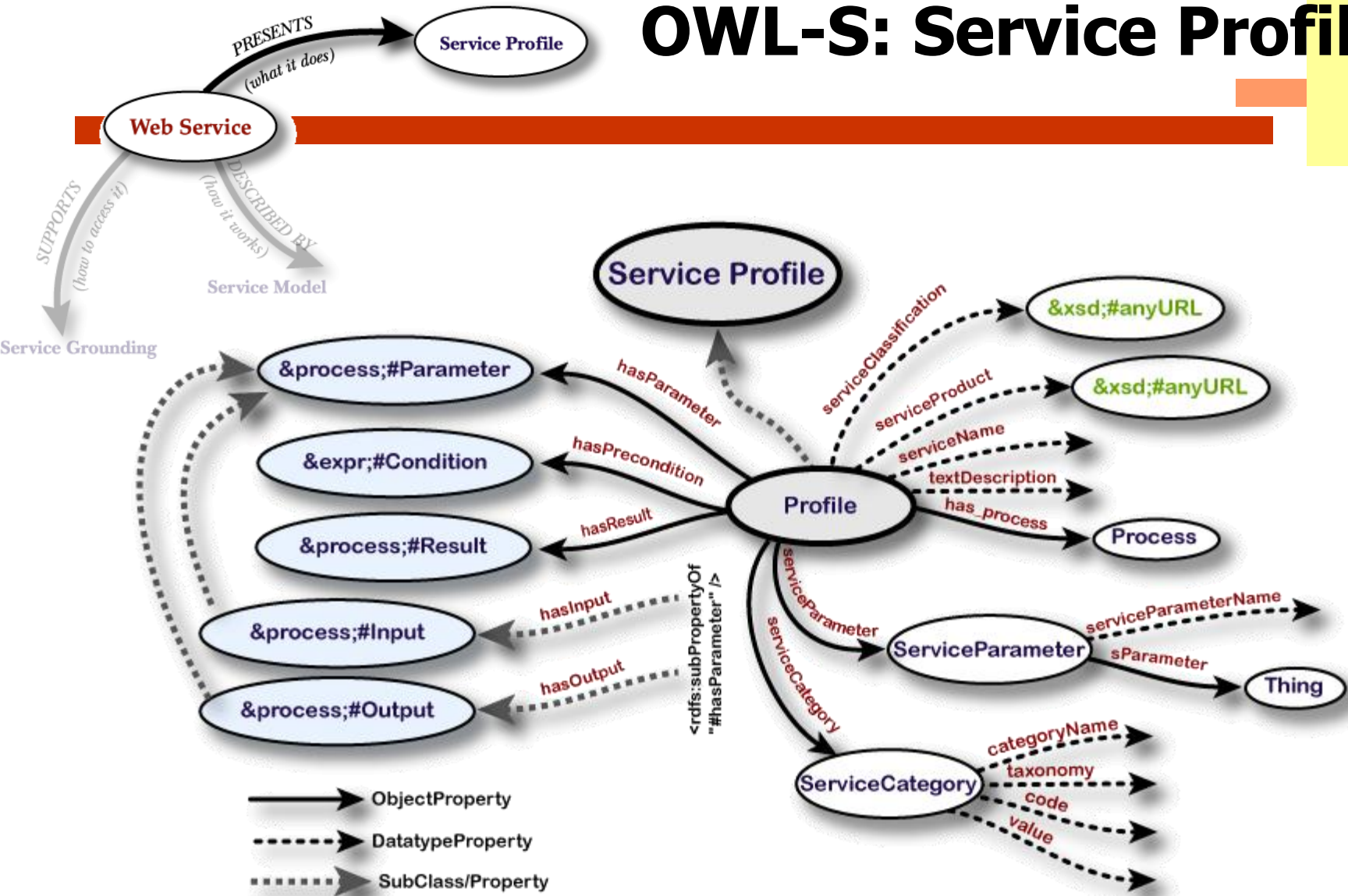
- communication protocol (RPC, HTTP, ...)
- marshalling/serialization
- transformation to and from XSD to OWL

- Control flow of the service
 - Black/Grey/Glass Box view
- Protocol Specification
- Abstract Messages

nach David Martin 2005

23.01.2012

OWL-S: Service Profile



nach David Martin 2005

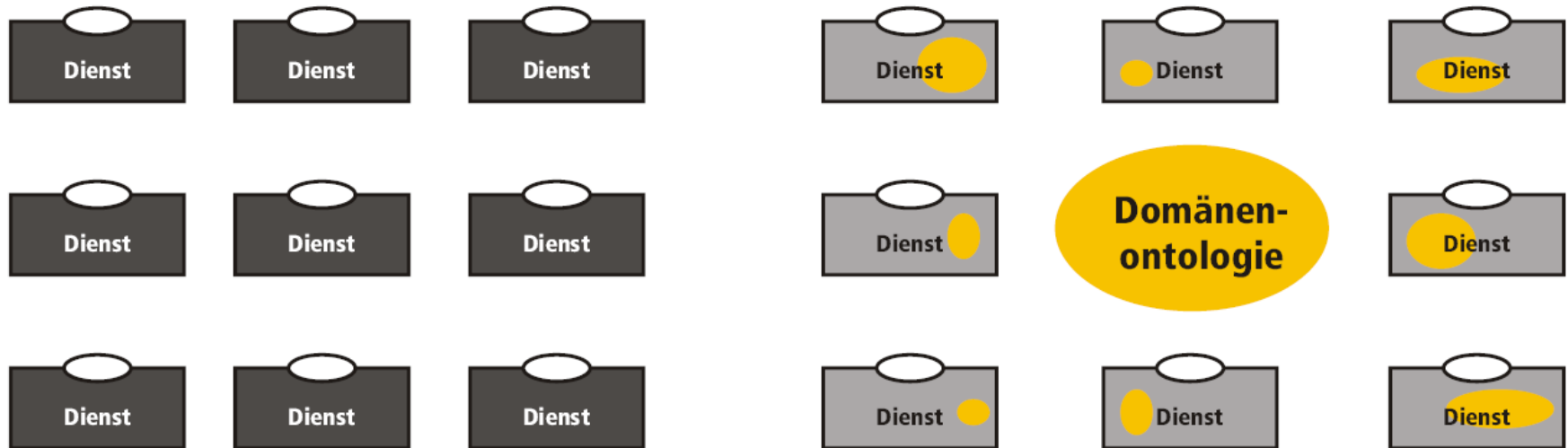
Ontologiezentrierte Entwicklung von dienstorientierten Architekturen



Anderer Ansatz: Ontologienzentrierter Architekturstil

- Tiefgehende semantische Dienstbeschreibungen, die auf Automatisierbarkeit von Dienstfindung und –komposition aus sind, sind sehr komplex.
- Allerdings können Ontologien auch ohne zuviel Formalität helfen, die semantische Kohärenz von Diensten zu erhöhen.
- Hierzu: gemeinsame zugrundeliegende Ontologie, auf die sich die Dienste beziehen
 - gewisser Verzicht auf Autonomie
 - diese Ontologie oder Auszüge daraus werden soweit sinnvoll während der Dienstausführung genutzt

Ontologiezentrierter Architekturstil (2)



herkömmliche dienstorientierte Architektur

Dienste veröffentlichen nur ihre Schnittstelle, die syntaktische Aufrufregeln spezifiziert.

ontologiezentrierte dienstorientierte Architektur

Zusätzlich zu den Schnittstellen ist über die Dienste bekannt, dass sie sich auf Teilaspekte der gemeinsamen Domänenontologie beziehen.

Beispiel: SOPRANO

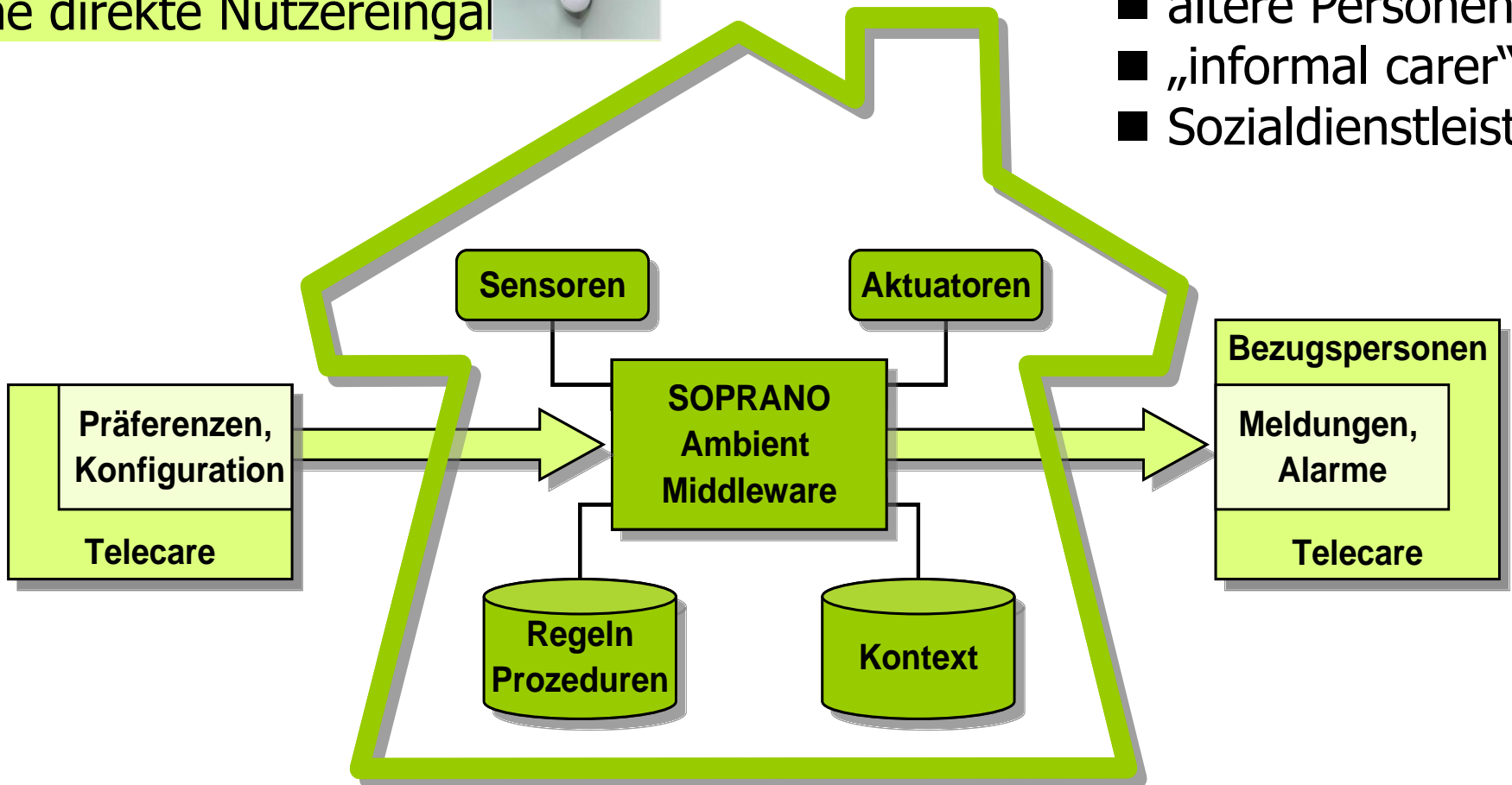
- SOPRANO ist ein Integrierendes EU-Projekt (IP), das sog. „Ambient-Assisted Living“-Lösungen mittels einer dienstorientierten Plattform ermöglichen will.
- Beispiele für Systemverhalten:
 - Wenn eine Person hinfällt, so wird abhängig vom Gesundheitszustand der Nachbar oder der Notarzt informiert.
 - Das System erinnert an Medikation und fällige Übungen.
 - Bei zu hoher/zu niedriger Temperatur wird automatisch das Fenster geschlossen/geöffnet oder die Heizung reguliert, um Auskühlung zu vermeiden.

System agiert vorwiegend
ohne direkte Nutzereingänge



Nutzer:

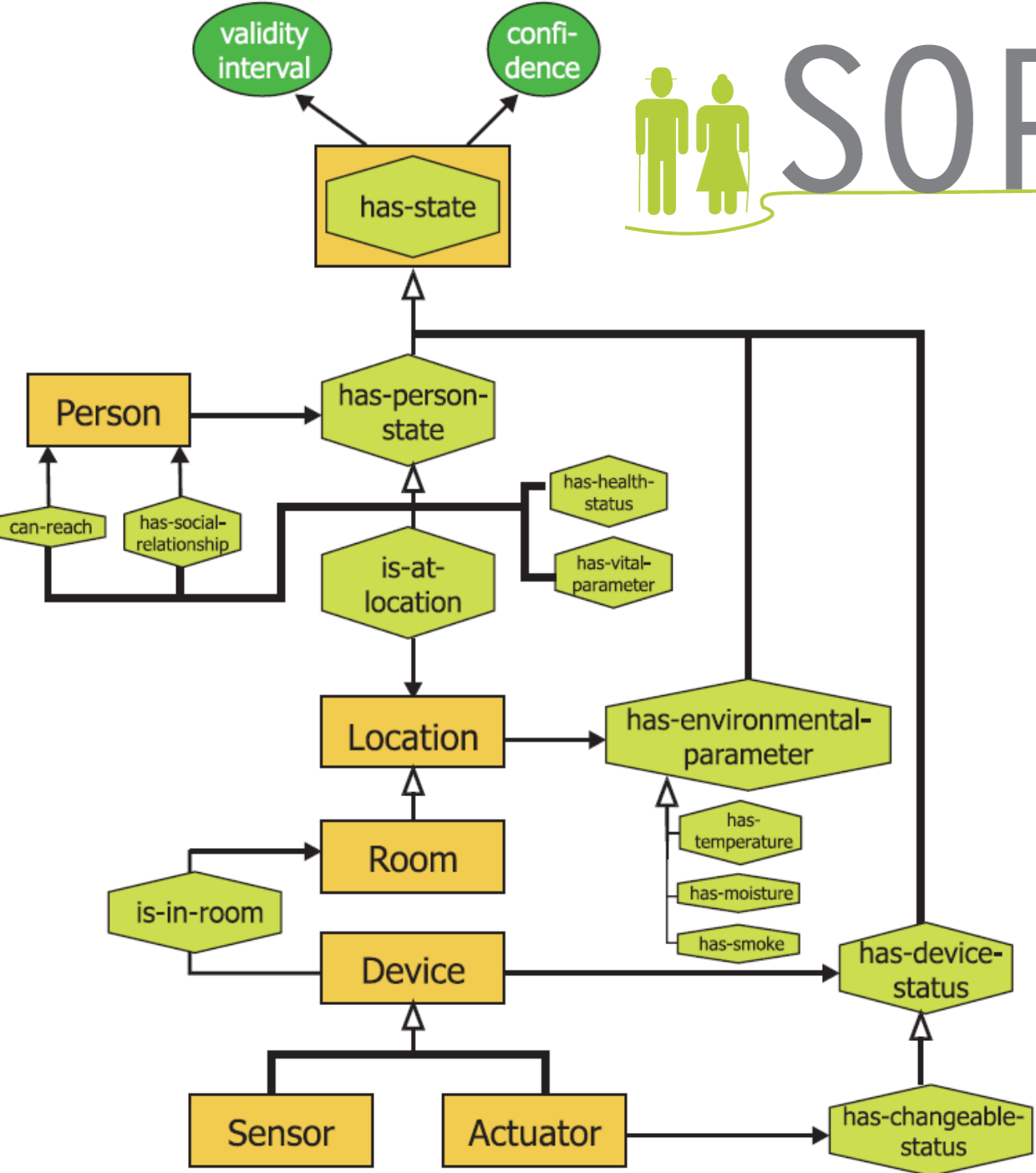
- ältere Personen
- „informal carer“
- Sozialdienstleister



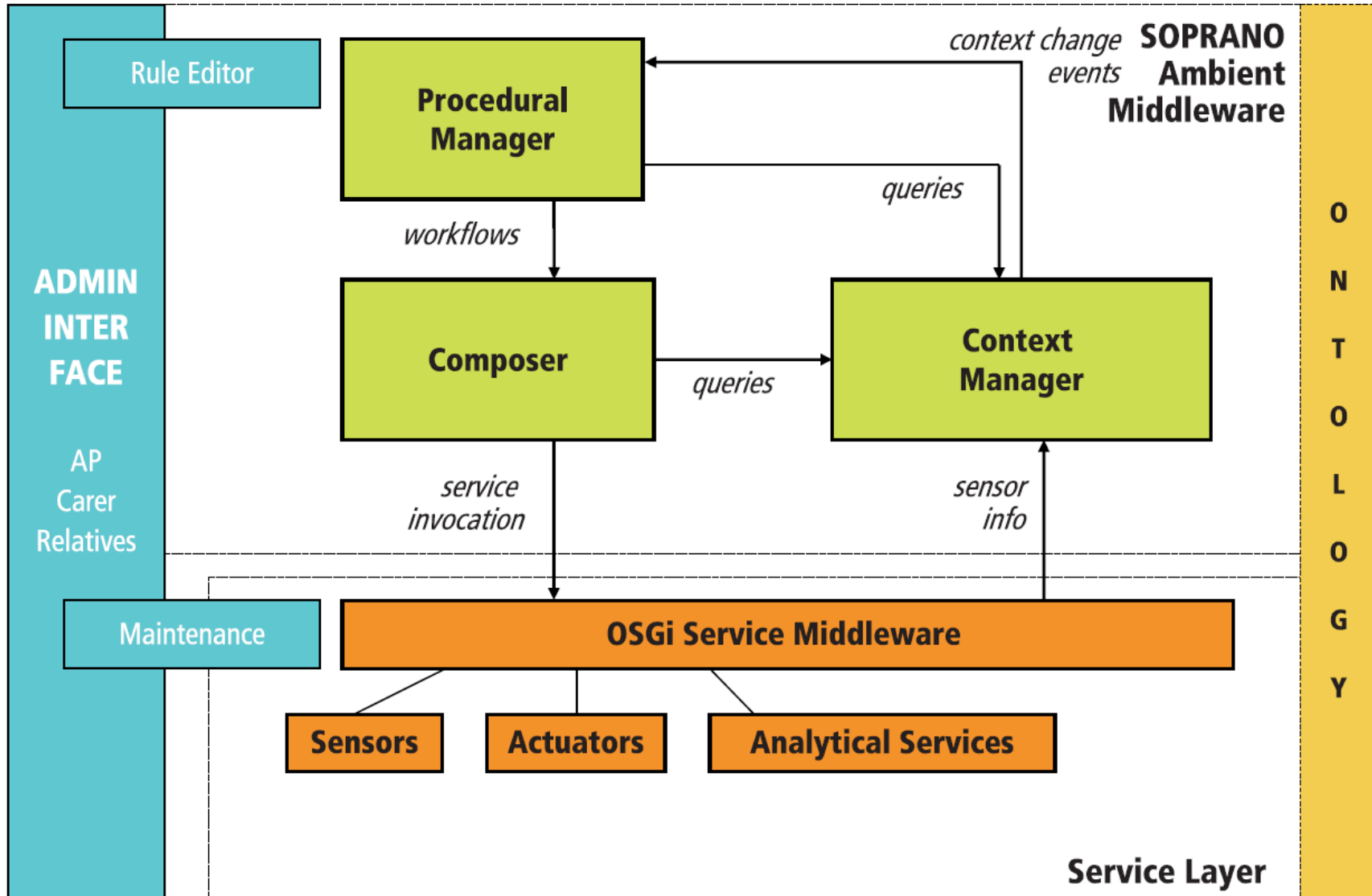
Semantik in SOPRANO



- SOPRANO erfordert ein Zusammenspiel von Sensoren, Aktuatoren und einer intelligenten Middleware.
- Idee: Ontologiezentrierte Entwicklung
 - dient zur Beschreibung der Sensoren und Aktuatoren
 - dient zur Beschreibung des Kontextes
 - dient als Vokabular für Abläufe

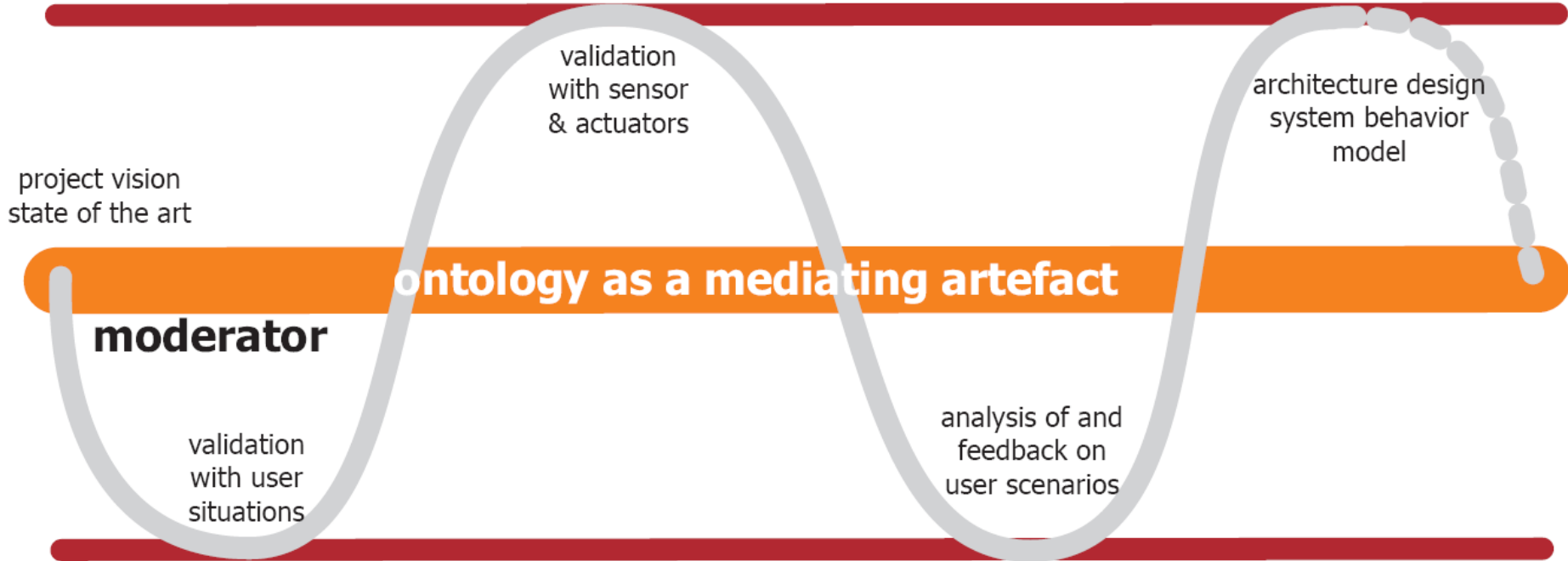


SOPRANO Architektur



Entwicklungsprozess

technology provider



moderator

application domain/users

increasing level of mutual and problem domain understanding

decreasing level of uncertainty

Fazit: Semantische Dienstbeschreibungen



Fazit: Semantische Dienstbeschreibungen



- Sind derzeit ein großes Forschungsthema
 - ist nicht auf „Dienste“ beschränkt, sondern kann natürlich auch zur semantischen Komponentenbeschreibung eingesetzt werden
 - von der Grundidee her ähnlich zu MDA-Ansätzen, aber meistens komplett andere Grundtechnologien
- Allerdings derzeit noch nicht praxistauglich
 - leichtgewichtige Ansätze wünschenswert

- BPEL4WS
<http://www.ibm.com/developerworks/library/ws-bpelcol1/>
- David Martin: Brief OWL-S Overview, 20.10.2005
http://ontolog.cim3.net/file/work/Ontolog-Discussion/Semantic-Web-Service-Ontology-Standard_20051020/OWL-S-BriefOverview--DavidMartin_20051020.ppt