

Es darf nur eine geben

# Semantik und Web Services: Beschreibung von Semantik

Wolfgang Dostal, Mario Jeckle, Werner Kriechbaum

Die Beschreibung von Bedeutungen verschiedener Art (von Verwaltungsinformation über Benutzungskosten bis hin zur Zusicherung einer festgelegten Dienstgüte) ist der zentrale Baustein des Semantic Web. Insbesondere in Service-orientierten Architekturen ist es notwendig Semantikbeschreibungen in von Maschinen verarbeitbarer Form bereitzustellen und gemeinsam mit dem Dienst anzubieten.

Der vorliegende zweite Teil der Artikelserie führt zunächst in die grundlegende Herausforderung der Beschreibung von Semantik ein und stellt mit dem Resource Description Framework (RDF) einen Ansatz zur maschinen-gerechten Darstellung von Aussagen über Semantik vor.

Im vorangegangenen Artikel [DoJe04] haben wir herausgearbeitet, dass Semantik ein wichtiger Bestandteil einer Service-orientierten Architektur (SOA) ist. In diesem Teil der Serie wird beschrieben, in welcher Form Semantik eine maschinenauswertbare Darstellung gegeben werden kann. Ein wichtiger Leitgedanke bei der Darstellung von Semantik ist – wie in den Grundideen des Semantic Web bereits gefordert –, dass sich ihre Beschreibung an die von Menschen gewohnte Art der Kommunikation anpassen soll.

Ist das Ziel die Nutzung von Semantik in der Kommunikation zwischen Maschinen, so kann nicht auf die intuitive Vorstellung von Semantik aufgebaut werden, die menschlicher Kommunikation zugrunde liegt. Die Herausforderungen beim Versuch, Semantik in einer für die Maschine auswertbaren Form darzustellen, sind vergleichbar denen eines Software-Designers bei der Erstellung eines Klassendiagramms. Die beim Design getroffenen Festlegungen hinsichtlich der Eigenschaften und Beziehungen einer Klasse sind für die Maschine während der Ausführung absolut verbindlich.

Eine Maschine ist – zumindest bis jetzt – nicht in der Lage, eine Festlegung (Aussage) nachträglich zu überarbeiten oder gar zu revidieren. Eine Klasse mit zu vielen Attributen – im Sinne von: für den gegebenen Kontext überflüssige – kann dazu führen, dass die ausgeprägten Objekte die Performanz der Anwendung negativ beeinflussen. Wohingegen eine Klasse mit zu wenigen Attributen – im Sinne von: für den

dies, dass Umfang und Niveau der Semantikbeschreibung an das zu lösende Problem angepasst sein müssen.

## Kommunikation

Das grundsätzliche Kommunikationsprinzip ist schematisch in Abbildung 1 dargestellt. Ein Sender möchte seine Informationen an einen Empfänger weitergeben. Dazu muss der Sender zuerst die zu übermittelnden Informationen codieren, d. h. er muss das, was sich in seinem Kopf (Vorstellungswelt) befindet, auf ein externes Medium transformieren. Im Rahmen dieses Umsetzungsprozesses werden interpretierte Informationen [WikiI] zu interpretationsbedürftigen „Daten“ [WikiD].

Anschließend erfolgt die Auswahl des zu verwendenden Kommunikationskanals, auf dem diese Daten als Nachricht übertragen werden können. Hierzu ist es notwendig, die Rohdaten zunächst in eine übertragbare Darstellung zu serialisieren. Der Empfänger nimmt die empfangene Nachricht entgegen, extrahiert daraus die Daten und de-

codiert sie, d. h. er bringt sie durch Interpretation in seine Vorstellungswelt und wandelt sie so wieder in Information um.

Zum erfolgreichen Austausch der Information ist es zudem notwendig, dass beide Seiten mit demselben Vokabular (Zeichenvorrat) und derselben Interpretation der Vokabeln arbeiten. Übersetzungen bilden hiervon keine Ausnahme, da in diesem Fall mindestens ein Kommunikationspartner mehrere Vokabulare interpretieren kann.

Das in Abbildung 1 dargestellte Kommunikationsschema praktizieren wir täglich, sobald wir ein Gespräch mit anderen Menschen führen, E-Mails austauschen oder aufgrund eines roten Ampelsignals stehen bleiben.\* In allen

\* Streng genommen ist dies keine Kommunikation, sondern eine Diagnose, da die Informationsübertragung nur in eine Richtung erfolgt.

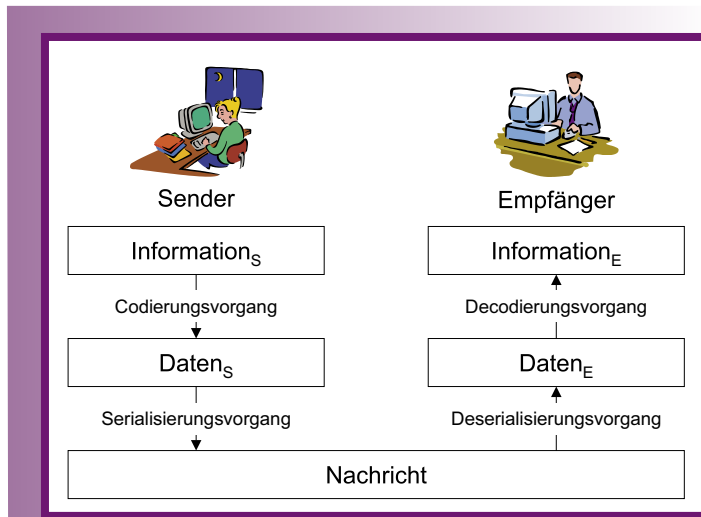


Abb. 1: Vereinfachtes Kommunikationsmodell für den Informationsaustausch



Fällen wird nicht die Bedeutung selbst übertragen, sondern lediglich eine Vokabel, die vom Empfänger interpretiert werden muss.

## Semantik

Glücklicherweise ist die Kommunikation zwischen Maschinen sowohl hinsichtlich des verwendeten Zeichenvorrates als auch der eingesetzten Übermittlungskanäle deutlich variantenärmer als ihr (zwischen-)menschliches Pendant, weshalb die eindeutige technische Beschreibung des für einen Kommunikationsvorgang relevanten Semantikausschnittes der realen Welt gelingen kann.

Nehmen wir als Beispiel nochmals die Ampel, deren Zeichenvorrat durchaus auch von Maschinen vollständig interpretiert werden kann. Was ist die Bedeutung eines roten Lichtes und wie erfolgt die Verknüpfung des roten Lichts mit seiner Bedeutung? Denn grundsätzlich wissen wir spätestens seit Aristoteles, dass Rot nichts „Stehen-bleiben-haftes“ an sich hat, also muss die Bedeutung „Rot heißt stehen bleiben“ erlernt worden sein.

Das rote Licht einer Ampel ist ein Zeichen, eine Dateneinheit, die übertragen wird. Information wird erst dann transferiert, wenn mit dem transportierten Zeichen ein Konzept verbunden ist, welches dem Empfänger bekannt ist und so – im Rahmen des Decodierungsvorganges – den eingegangenen Daten zugeordnet werden kann.

So ist die Farbe (das Zeichen) *Rot* für uns mit einem Konzept für ein *Stehen-bleiben* verbunden. Wohingegen das Zeichen *Um90Cg* für die meisten Empfänger keine Bedeutung beinhaltet.\*\* Semantik ergibt sich also aus der Verknüpfung eines Signals mit einem Konzept, das etwas beschreibt. Das Zeichen *Rot* ist verknüpft mit dem Konzept des „Stehen-bleibens“.

Typischerweise kann dieser Interpretationsvorgang nicht in allen Fällen durch den technischen Empfänger der Nachricht geleistet werden. Lediglich in Fällen, in denen die Daten so eindeutig formuliert sind, dass kein Interpretationsspielraum bleibt, kann die Deserialisierung bereits hinreichend zur Informationswiedergewinnung sein. Die Herausforderung im Umfeld der Service-orientierten Architekturen besteht daher darin, Semantik so eindeutig formulieren zu können, dass menschliche Interpretationsvorgänge verzichtbar werden, ohne Semantik zu verlieren.

Mit dieser Aussage befinden wir uns jedoch in einem Dilemma, da Semantik nicht niedergelegt, sondern nur verstanden werden kann. Auch Menschen können eigentlich lediglich ein bekanntes Zeichen mit einem erlernten

Konzept verknüpfen, aber nicht direkt aus dem Zeichen auf eine zugeordnete, aber möglicherweise unbekannte, Semantik schließen.

Zudem stellen natürliche Sprachen keinen hinreichenden Zeichenvorrat zur Verfügung, um alle denkbaren Konzepte so eindeutig zu benennen, dass alle Interpretationslücken geschlossen werden können. Beispielsweise treten in der Realität Homonyme (dasselbe Zeichen für verschiedene Bedeutungen) und Synonyme (dieselbe Bedeutung wird mittels verschiedener Zeichen bezeichnet) auf, die neben lustiger Unterhaltung (Stichwort: Teekesselchen) auch zu jeder Menge manuell durch den menschlichen Kommunikator aufzulösender Verwirrung führen.

Ausgeschlossen werden kann diese Gefahr im Umfeld maschineller Kommunikation nur, indem Konzepte nicht durch geläufige Zeichen, sondern in weltweit eineindeutiger Weise benannt werden. Ein in diesem Sinne verwendeter Name kann als eindeutige Referenz auf das Konzept verstanden werden. Eine solche Referenz könnte beispielsweise ein Uniform Resource Locator (URL) sein.

Allerdings ist die URL als technisches Vehikel mit dem Nachteil behaftet, dass der durch sie bezeichnete Endpunkt der Referenz eine physisch existierende Ressource (HTML-Seite, Binärdatei, etc.) im Web sein muss.

## Von der URL zur URI

Um diese Einschränkung zu vermeiden, wurden URLs zu Uniform Resource Identifiern (URI) verallgemeinert. Konstrukte dieses Typs dienen ausschließlich der Identifikation, nicht mehr jedoch der Lokalisierung einer Ressource. Im Kern ist der Unterschied zwischen URL und URI mit dem zwischen Adresse (Anschrift, die Wohnort eindeutig bezeichnet) und Namen (der den Wohnhaften eindeutig bezeichnet ohne Rückschlüsse auf seinen Wohnort zuzulassen) vergleichbar.

Syntaktisch besteht jede URI aus drei Teilen:

```
<schema>:<schema-specific-part>#<fragment>
```

Mit Hilfe des `schema`s kann ein Namensraum (d. h. Gültigkeitsbereich) definiert werden, für den die URI Wirksamkeit besitzen soll. Im Web finden zumeist die Schemata „http“, „ftp“ und „mailto“ Anwendung. Innerhalb des Gültigkeitsbereiches eines Schemas kann durch den `schema-specific-part` ein beliebiger bestimmter Teil bezeichnet werden. Durch diesen Anteil kann der URI-Verwalter seinen Namensraum frei unterstrukturieren.

Alle nach dem separierenden Nummernsymbol (#) angegebenen Teile werden relativ zum schemaspezifischen Anteil interpretiert. Bekannteste Verwendung hierfür dürften die Fragmentidentifikatoren zur Adressierung

von Teilen einer HTML-Seite sein. Aufgrund der Interpretation der URI als Namen eines Konzeptes – und nicht als Verweis auf eine physische Ressource – muss eine URI nicht zwingend auflösbar sein.

Dasselbe gilt (verwirrenderweise) auch, wenn URLs wie Web- oder E-Mail-Adressen als URI Verwendung finden ... Dennoch erfreuen sich URLs als URIs großer Beliebtheit, da ihre einfache Syntax gut verstanden wird und die Kopplung zentraler und dezentrale Vergabemechanismen ihre Eindeutigkeit gewährleistet.

Das Paradebeispiel für den Bedarf des URI-Konzepts in der Web-Services-Welt ist die UDDI-Komponente. Die UDDI-Registry ist ein Verbund autonom agierender UDDI-Betreiber, die ihre Einträge in festgelegten zeitlichen Abständen gegenseitig abgleichen. Jedes Unternehmen, das einen Service bei einem der UDDI-Anbieter registriert, erhält einen eindeutigen Schlüssel zugewiesen, der das Wurzelement `businessEntity` des UDDI-Eintrages und damit den eingetragenen Dienstanbieter identifiziert. Bei dem Replikationsvorgang darf dieser natürlich nicht mit einem Schlüssel aus einer der anderen UDDI-Instanz kollidieren. Da die Zuteilung von Wertebereichen zur Schlüsselbildung an jeden UDDI-Betreiber zu unflexibel ist, wird zur Erzeugung der Schlüssel ein Algorithmus (ISO/IEC 11578:1996) eingesetzt, der mit mathematischen Mitteln hinreichend sicherstellt, dass alle unabhängig voneinander erzeugten Schlüssel dennoch eindeutig sind. Die so erzeugten Schlüssel werden Universally Unique Identifier (UUID) genannt.

## Semantikbeschreibung mit dem Resource Description Framework

Um das Ziel der Automatisierung in einer SOA zu erreichen, ist das URI-Konzept alleine noch nicht ausreichend. Mit einer URI kann „lediglich“ ein Zeichen (oder eine Zeichenfolge) mit einem Konzept verknüpft werden. Für eine Anwendung ist es jedoch auch wichtig zu beschreiben, ob und wie verschiedene Konzepte zusammenhängen und wechselwirken. So kann es gewünscht sein, die Beschreibung eines Service zu seiner Leistungsbeschreibung in Beziehung zu setzen. Diese wiederum muss, damit sie maschinenverwertbar ist, ebenfalls durch ein Zeichen repräsentiert sein. Die URI wiederum stellt die Verknüpfung mit dessen Definition (Konzept) her. Darüber hinaus muss auch die Beziehung zwischen den beiden Konzepten eindeutig und maschinenlesbar bestimmt werden. Es wird daher eine technische Möglichkeit benötigt, um in maschinenauswertbarer Form Beziehungen zwischen Konzepten darstellen zu können.

Diesen etwas kniffligen Absatz werden wir nun an einem Beispiel verdeutlichen. Der Nachrichtenanbieter `example.com` stellt einen

\*\* Von der leeren Informationseinheit als Spezialfall einer Informationsübertragung wollen wir hier absehen.

Dienst zur Verfügung, über den Kunden den aktuellen Börsenkurs in Echtzeit abrufen können. Dieser Dienst wird unter der Bezeichnung **Stock Quote Service** als Web Service angeboten. Mit diesem Service sind bestimmte vertragliche Bedingungen verknüpft. Jeder Vertrag ist durch eine eindeutige Nummer (Contract ID) bestimmbar.

Aus dieser Beschreibung sollen nun die wichtigen Informationen extrahiert werden, die den Dienst beschreiben. Oder anders ausgedrückt, wir bilden Aussagen über den Service:

- (a) Der Dienst „Stock Quote Service“ ist ein Service, der Daten in Echtzeit liefert.
- (b) Der Dienst „Stock Quote Service“ ist definiert durch einen Vertrag.
- (c) Der Dienst „Stock Quote Service“ hat eine WSDL-Beschreibung.
- (d) Die WSDL-Beschreibung (also der Service) ist mit einem Vertrag verknüpft.
- (e) Der Vertrag zum Service „Stock Quote Service“ hat eine Vertragsnummer.

Alle weiteren Informationen über diesen Dienst erklären wir für die Automatisierung erst einmal für irrelevant und lassen sie aus Übersichtlichkeitsgründen außen vor. Trotz der Einfachheit dieser Aussagen ist zu erkennen, dass die menschliche Sprache eine Komplexität besitzt, die nur schwer durch einen Formalismus einzufangen ist (siehe Aussage a bzw. d).

Alle oben aufgeführten Aussagen des Beispiels können in drei Teile zerlegt werden, wobei die Aussage (d) noch weiter vereinfacht werden muss. Die Aussage (c) stellt sich somit wie folgt dar:

{Der Dienst " Stock Quote Service " /hat/eine WSDL-Beschreibung. }

Diese Darstellung zeigt, dass eine Aussage in ihrer einfachsten Form mittels eines Tripels dargestellt werden kann. Die Elemente des Tripels werden je nach Anwendungsgebiet (Domäne) unterschiedlich bezeichnet.

In der flexiblen Bezeichnungsmöglichkeit der Elemente zeigt sich gleichermaßen eine Stärke und eine Schwäche des Tripel-Konzepts. Jedes Anwendungsgebiet, das Beziehungen (property) zwischen verschiedenen Objekten (resource) ausdrücken muss, kann seine eigene Terminologie definieren, d. h. jede Domäne kann das Tripel-Konzept sprachlich problemlos in sein Wissensgebäude aufnehmen. Allerdings kann es dann mitunter schwierig werden, wenn sich Vertreter verschiedener Domänen über ein Tripel unterhalten möchten.

So! Und damit sind wir mittendrin im Resource Description Framework (RDF) des W3C. Denn dieses liefert einen standardisierten Formalismus zur Abbildung von Aussagen über beliebige Ressourcen (d. h. Personen, Web-Seiten, Web Services oder Web-ferne Konzepte).

Da eine rein textuelle Darstellungen, wie oben gezeigt, wenig benutzerfreundlich (insbesondere wenn der Leser eine Maschine ist) und die Erstellung eines Vokabulars auf diesem Wege fehleranfällig ist (insbesondere für Menschen), wurden formale Darstellungsformen entwickelt. Zum einen ist dies die so genannte N3-Darstellung, die eine formale mathematische Bearbeitung von Aussagen erlaubt. Diese Darstellung erfreut sich vor allem innerhalb des W3Cs großer Beliebtheit. Eine weitere Möglichkeit ist es, Aussagen graphisch darzustellen, was sich bei den Anwendern in der IT durchgesetzt hat.

Die textuellen Aussagen (a) bis (e) können wie in Abbildung 2 als gerichtete Graphen dargestellt werden. Die elliptischen Knoten der Darstellung werden im RDF als **Ressourcen** bezeichnet, zwischen denen eine gerichtete Beziehung (Kante) existiert.

Alle Darstellungsformen (Text, Graph, N3) haben gemeinsam, dass die verwendeten Symbole mit einer URI hinterlegt werden können (s. Abb. 3), damit die Daten, wie im vorangegangenen Abschnitt gezeigt, eindeutig auswertbar werden. Eine Ausnahme bilden Konstanten, die nicht durch eine externe Definition beschrieben werden (dargestellt durch Rechteckssymbole).

Zur Darstellung von Aussagen höherer Komplexität kann jede Ressource beliebig viele Beziehungen zu sich selbst oder anderen Ressourcen besitzen. Selbst Aussagen höherer Ordnung, mithin Aussagen über Aussagen, können formuliert werden (reification).

Als – zugegebenermaßen auf den ersten Blick etwas unorthodoxes – RDF-Anwendungsbeispiel stellen wir daher unter [Rotkäppchen] wichtige Aussagen des Märchens *Rotkäppchen* als RDF-Datei zur Verfügung.

### RDF/XML

Innerhalb der W3C-Arbeitsgruppe spielt die Serialisierung eines RDF-Graphen in ein XML-Dokument (das Format wird dann als „RDF/XML“ bezeichnet) eine untergeordnete Rolle. Ein Grund könnte sein, dass XML-Dokumente die Eigenart besitzen, ihren Umfang sehr schnell anwachsen zu lassen, was nicht unbedingt zu ihrer Lesbarkeit für Menschen beiträgt. Allerdings sind diese XML-Dokumente außer zu Kontrollzwecken auch nicht zur Lektüre für Menschen gedacht. Hinzu kommt, dass die XML-Schemabeschreibung, die der XML-Darstellung von RDF zugrunde liegt, es erlaubt, eine Ressource oder Bezie-

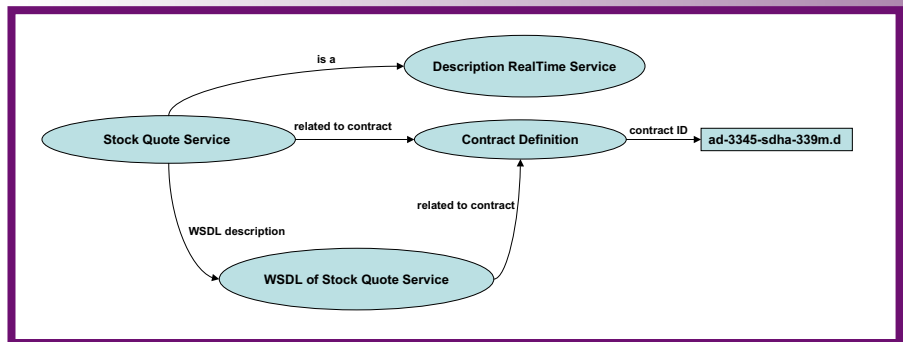


Abb. 2: Das Beispiel „Stock Quote Service“ mit sprechenden Bezeichnungen (Zeichen)

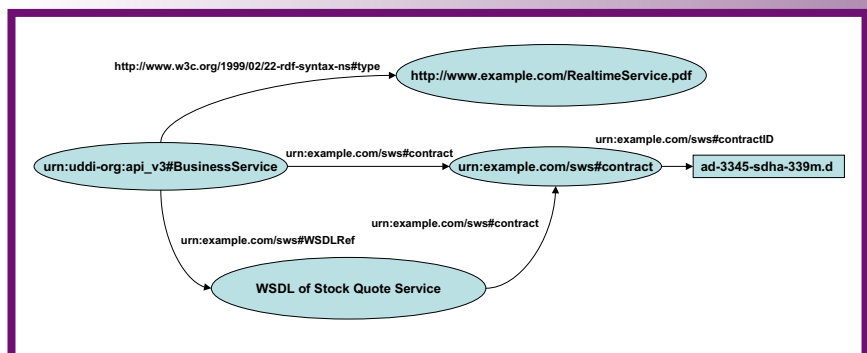


Abb. 3: Das Beispiel „Stock Quote Service“ mit maschinenauswertbaren Bezeichnungen (URI)



hung wahlweise als Element oder Attribut darzustellen. Dieselbe Information kann somit in unterschiedlicher Form dargestellt werden, was den Wiedererkennungseffekt verringert.

In der Praxis erfreuen sich die XML-Notation und die graphische Darstellungsform großen Zuspruchs. Ein Grund ist sicher, dass die Darstellung in XML nicht auf einem abstrakten Niveau einer Algebra, wie die alternative N3-Notation, ausgedrückt wird. Auch die Verfügbarkeit leistungsfähiger graphischer XML-Editoren sowie RDF-spezifischer angepasster APIs [Jena] und graphischer Werkzeuge zur manuellen Erstellung und automatisierten Generierung von RDF/XML-Dokumenten dürfte wesentlich zur Beliebtheit der XML-Darstellung beitragen. Für den Einsatz von XML/RDF im Kontext Service-orientierter Architekturen spricht zudem, dass XML bereits im Web-Services-Umfeld allgegenwärtig ist (z. B. SOAP, WSDL, ...) und auf dieser Basis kein Bruch in den verwendeten Techniken erfolgt.

Die elementaren Merkmale von RDF/XML werden wir im Folgenden vorstellen. Dazu betrachten wir die XML-Serialisierung des Graphen aus Abbildung 3, die im Listing 1 wiedergegeben ist. Das Wurzelement eines RDF-Dokuments besitzt immer die Form:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
...
</rdf:RDF>
```

Weitere Namensräume können gemäß der Regeln zur Definition von Namensräumen [XMLNS] angegeben werden. Die Nutzung von Namensräumen

war in früheren RDF-Spezifikation optional. Da die Namensräume jedoch auch Bestandteil der konzeptidentifizierenden URIs sind, wurde die Nutzung von Elementen ohne Namensräume als „veraltet“ (deprecated) erklärt.

In unserem Beispiel folgt als erstes Kindelement von `rdf:RDF` die Beschreibung einer Ressource:

```
<rdf:Description>
...
</rdf:Description>
```

Jede im RDF-Dokument aufgeführte Ressource wird durch ein Element `rdf:Description` inhaltlich definiert. Verwirrend ist hierbei die Möglichkeit, dass Ressource-Beschreibungen als Kinder eines beliebigen Elements des RDF-Dokuments dargestellt werden können, und das ohne Veränderung der semantischen Bedeutung der Ressource. Im Beispiel aus Listing 1 sind die Ressourcen `urn:uddi-org:api_v3#BusinessService`, `urn:example.com/sws#contract` und `http://www.example.com/stockQuote.wsdl` auf verschiedenen Hierarchiestufen angesiedelt. In allen Fällen wurden Ressourcen definiert, die im gesamten RDF-Dokument sichtbar sind.

Die Zuordnung einer URI zu einer Ressource erfolgt mit Hilfe des Attributes `rdf:about`. Über diese URI ist diese Ressource dann für andere referenzierbar. Wird einer Ressource keine URI zugeordnet, spricht man von einer anonymen Ressource.

In unserem Beispiel ist der „Stock Quote Service“ eindeutig durch die URI `urn:uddi-org:api_v3#BusinessService` identifiziert. Durch die Wahl der URI will der Service-Anbieter (hier die Autoren) ausdrücken, dass es sich um einen `BusinessService` im Sinne der UDDI-Spezifikation in der Version 3 handelt.

Allerdings handelt es sich hierbei um eine Meta-Information, die einer Maschine nur dann zugänglich ist, wenn die Semantik dieser URI irgendwo (bekannt und erreichbar) beschrieben wurde. Für den Dienstanbieter ist in diesem Zusammenhang nur wichtig, dass der Service eindeutig für Mensch und Maschine identifizierbar ist. Aufgrund der Wahl eines sprechenden Namens ist es einem Menschen evtl. möglich, die Semantik des Services zu erschließen, ohne die Spezifikation zu lesen. Die Frage, wie dieser Service gefunden wird, ist Thema des vierten Teils dieser Serie; hier ist nur die Eindeutigkeit wichtig.

Betrachten wir nun, wie Beziehungen (property) von Ressource (Subjekt) zu Ressource (Objekt) beschrieben werden. Wird eine Ressource innerhalb einer anderen Ressource definiert:

```
<rdf:Description rdf:about="urn:uddi-org:api_v3#BusinessService">
  <sws:WSDLRef>
    <rdf:Description rdf:about="http://www.example.com/stockQuote.wsdl">
      ...
    </rdf:Description>
  </sws:WSDLRef>
</rdf:Description>
```

dann wird die umschließende Ressource `urn:uddi-org:api_v3#BusinessService` als Start und die umschlossene Ressource `http://www.example.com/stockQuote.wsdl` als Ende der (gerichteten) Beziehung interpretiert. Das Element `sws:WSDLRef` definiert die Beziehung der beiden Ressourcen, die vollständig expandiert `urn:example.com/sws#WSDLRef` lautet. Dazu wurde in der RDF-Spezifikation festgelegt, dass jedes Kindelement von `rdf:Description` als Beziehung interpretiert wird, falls es sich nicht um ein Element aus dem RDF-Vokabular handelt.

Dieser Ansatz ist jedoch für komplexere Beziehungen, wie beispielsweise Mehrfachbeziehungen zu einer Ressource, nicht ausreichend. Deshalb wurde das Attribut `rdf:resource` in der Spezifikation definiert. Damit ist es möglich eine Beziehung zu einer beliebigen nicht-anonymen Ressource im RDF-Dokument zu definieren. In unserem Beispiel wird die Ressource `http://www.example.com/stockQuote.wsdl` über den Ausdruck

```
<sws:contract rdf:resource="urn:example.com/sws#contract"/>
```

mit der Ressource `urn:example.com/sws#contract` verknüpft. Die Beziehung lautet in diesem Fall vollständig expandiert `urn:example.com/sws#contract`.

In den vorangegangenen Beispielen wurden die Beziehungen jeweils vom Dienstanbieter gemäß seinen Anforderungen domänenspezifisch definiert. Das XML-Schema für RDF hält jedoch bereits einige vordefinierte Beziehungstypen,

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:uddi="urn:uddi-org:api_v3#"
  xmlns:sws="urn:example.com/sws#">
  <rdf:Description rdf:about="urn:uddi-org:api_v3#BusinessService">
    <rdf:type rdf:resource="http://www.example.com/RealtimeService.pdf"/>
    <sws:WSDLRef>
      <rdf:Description rdf:about="http://www.example.com/stockQuote.wsdl">
        <sws:contract rdf:resource="urn:example.com/sws#contract"/>
      </rdf:Description>
    </sws:WSDLRef>
    <sws:Contract>
      <rdf:Description rdf:about="urn:example.com/sws#contract">
        <sws:ContractID>
          rdf:datatype="xsd:string">ad-3345-sdha-339m.d
        </sws:ContractID>
      </rdf:Description>
    </sws:Contract>
  </rdf:Description>
</rdf:RDF>
```

Listing 1: XML-Serialisierung des Graphen aus Abb. 3



nebst einer durch die W3C-Spezifikation festgelegten Semantik, bereit. Im Beispiel des Listings 1 wird die Beziehung zwischen dem Subjekt `urn:uddi-org:api_v3#BusinessService` und dem Objekt `http://www.example.com/RealtimeService.pdf` mit Hilfe des Elements `rdf:type` ausgedrückt. Damit wird beschrieben, dass es sich bei dem angebotenen Service „Stock Quote Service“ um einen Echtzeit-Service handelt. Die genaue Definition, was ein Echtzeit-Service ist, befindet sich in dem pdf-Dokument, das mit Hilfe der URI `http://www.example.com/RealtimeService.pdf` definiert ist.

Sollte diese URI darüber hinaus auch eine URL sein, dann kann man dort nachlesen was ein RealtimeService ist. Solange jedoch Maschinen die Texte in pdf-Dokumenten nicht verstehen können, muss das Nachlesen durch einen Menschen erfolgen. Sollte ein anderer seinen Service mit dieser URI verbinden, obwohl es sich nicht um einen Echtzeit-Service handelt, kann eine Maschine – wie oben unter „Semantik“ bereits beschrieben – dies nicht erkennen.

## Zusammenfassung

Der zweite Teil unserer Serie stellte die grundlegenden Anforderungen der Semantikdarstellung im Umfeld maschineller Kommunikation, unter besonderer Berücksichtigung Service-orientierter Architekturen, dar.

Die Kombination der Techniken der URI und des RDF-Vokabulars weisen den Weg bekannte Konzepte, zu deren Semantik bereits eine Übereinkunft existiert, in eindeutiger Weise zu benennen und flexibel miteinander in Beziehung zu setzen. Die sinngebende Instanz ist nach wie vor der Mensch, der die Semantik codiert.

Im nächsten Artikel werden wir anhand von RDF-Schema zeigen, wie die Ausdrucksmöglichkeiten zur Beschreibung von Semantik weiter ausgebaut und verfeinert werden können. Des Weiteren werden wir zeigen, wie lokal definierte Semantiken mit Hilfe von Ontologien (Stichwort: DAML, OWL) untereinander verknüpft werden können, um den SOA-Gedanken „think global, act local“ – lose Kopplung – verwirklichen zu können.

## Literatur und Links

**[DoJe04]** W. Dostal, M. Jeckle, Semantik, Odem einer Service-orientierten Architektur, in: JavaSPEKTRUM, 1/04

**[Jena]** Jena – A Semantic Web Framework for Java, <http://jena.sourceforge.net>

**[Rotkäppchen]** RDF-Anwendungsbeispiel, <http://www.jeckle.de/semanticWebServices/rotkaeppchen/>

**[WikiD]** Wikipedia, die freie Enzyklopädie, Stichwort: Daten, <http://de.wikipedia.org/wiki/Daten>

**[WikiI]** Wikipedia, die freie Enzyklopädie, Stichwort: Information,

<http://de.wikipedia.org/wiki/Information>

**[XMLNS]** T. Bray, D. Hollander, A. Layman (Hrsg.), Namespaces in XML, W3C Recommendation, 1999, <http://www.w3.org/TR/REC-xml-names>



**Dr. Wolfgang Dostal** ist für die IBM Global Services (AMS SI) als IT-Architekt tätig. Er befasst sich intensiv mit Internettechnologien und Anwendungsintegration (EAI). Der Focus seiner Arbeit liegt auf der Erstellung von J2EE-Anwendungen und der Einführung von Web Services. An der Fachhochschule Mainz unterrichtet er zum Thema objektorientierte Softwareentwicklung. E-Mail: [wolfgang.dostal@de.ibm.com](mailto:wolfgang.dostal@de.ibm.com).

**Mario Jeckle** hat eine Professur für Software Engineering an der Fachhochschule Furtwangen inne und arbeitet schwerpunktmäßig im Umfeld XML und Datenmodellierung. Als Mitglied des Advisory-Committees vertritt er die DaimlerChrysler-Forschung im W3C. E-Mail: [mario@jeckle.de](mailto:mario@jeckle.de).

**Werner Kriechbaum** arbeitet als IT-Architekt im Entwicklungslabor der IBM Deutschland. Schwerpunkte seiner Tätigkeit sind Performanz-Analyse und Optimierung von Anwendungen und die Beschreibung und Strukturierung komplexer Daten für Multimedia-Archive und biowissenschaftliche Anwendungen. E-Mail: [kriechba@de.ibm.com](mailto:kriechba@de.ibm.com).