In: Klaus-Peter Fähnrich; Stefan Kühne; Andreas Speck; Julia Wagner (Hrsg.): Integration betrieblicher Informationssysteme: Problemanalysen und Lösungsansätze des Model-Driven Integration Engineering. Eigenverlag Leipziger Informatik-Verbund (LIV), September 2006, ISBN-10: 3-934178-66-9, ISBN-13: 978-3-934178-66-3.

Mercedes Car Group (MCG) Enterprise Architektur – Ein Ansatz zur semantischen Modellierung der Services in einer SOA

Michael Herrmann

DaimlerChrysler AG
Fronäckerstraße 40, 71059 Sindelfingen, Germany
E-Mail: michael.hm.herrmann@daimlerchrysler.com

Muhammad Ahtisham Aslam

Universität Leipzig, Institut für Informatik, Abteilung Betriebliche Informationssysteme
Augustusplatz 10–11, 04109 Leipzig, Germany
E-Mail: aslam@informatik.uni-leipzig.de

Kurzfassung

Mit der Serviceorientierten Architektur (SOA) sollen Systeme für die Zukunft gebaut werden. Diese Vision soll in Form von Implementierungen basierend auf echten Projektanforderungen verifiziert werden. Die Projektstruktur besteht aus einem koordinierenden Projekt mit strategischer Ausrichtung sowie zwei weiteren Projekten mit zu realisierenden Anforderungen.

Das übergreifende Projekt hat das Ziel, die SOA-Projekte zu koordinieren sowie den fehlenden Raum für die Evaluierung neuer Konzepte, durch zusätzliche Implementierungen, zu ergänzen. Das primäre Projektziel ist die Verifizierung der kompletten SOA-Vision (inkl. Technologie, Organisation, Methoden) anhand echter Anforderungen.

In diesem Artikel werden das übergreifende Projekt und eines der beiden o. g. SOA-Projekte skizziert sowie die Basis der semantischen Integration in eine SOA näher erläutert. Die Hypothese „Konzepte, Relationen und Propositionen sind für die Definition eines formalen Systems ausreichend, um durch Inferenz äquivalente Services zu erkennen“ wird, basierend auf einer echten Projektanforderung, in einen Laborversuch verifiziert und validiert.

1. Motivation

SOA beherrscht gegenwärtig die Diskussion zur Gestaltung unternehmensweiter IT-Landschaften und wird als Paradigmenwechsel der heutigen Informationstechnologie dargestellt. Kern der Diskussion ist die Komplexität der IT zu reduzieren, die Agilität der Unternehmen zu unterstützen und die Wiederverwendung zu vereinfachen. Der schnelle Wandel in der IT führt unter anderem zu einem permanenten Anpassen und Erweitern von bereits eingesetzten Lösungen. Hinzu kommen neu entwickelte Lösungen, die den derzeit geltenden Erkenntnissen entsprechen. Betriebswirtschaftliche Faktoren, Unternehmensfusionen sowie die Globalisierung und die Adaption der Geschäftsprozesse durch das Einbinden der Kunden und Lieferanten, verstärken diesen Wandel. Das Zusammentreffen geschäftsprozessorientierter und informationstechnologischer Interessen ist unvermeidbar. Durch nicht voraussehbare Änderungen in der Zukunft sollen IT-Systeme basierend auf einer SOA miteinander verbunden werden, um schneller an neue Bedingungen angepasst werden zu können. Die SOA ermöglicht eine iterative Vorgehensweise und ist somit ein mögliches Mittel zur weichen Migration. Dieser Anspruch an eine SOA soll in Form

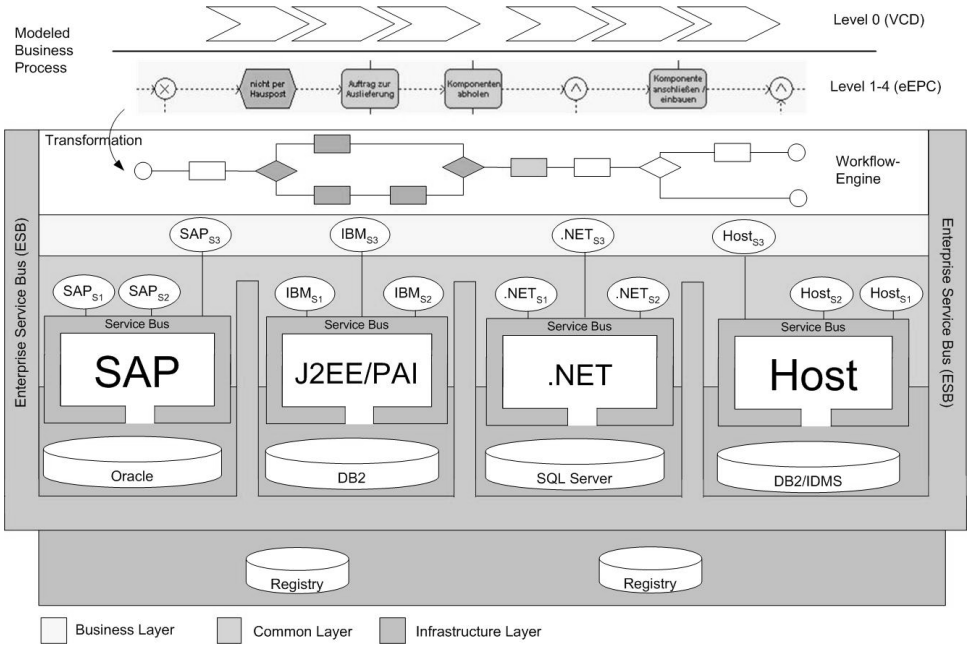

Abbildung 1: Die SOA-Vision.

von echten Projektanforderungen/Projektszenarien in einem Unternehmen geprüft werden.

2. Verifizierung der SOA-Vision

Ein einzelnes Projekt, das die Anforderungen des Fachbereichs realisiert, kann die Evaluierung der kompletten SOA-Vision (siehe Abbildung 1) nicht leisten. Erschwerend kommt hinzu, dass ein lokaler Nutzen nur schwer in einem ersten „SOA-Projekt“ aufgezeigt werden kann. Somit übernimmt das übergreifende Projekt exemplarisch, durch operatives Handeln, das Schließen der Lücken um aussagekräftige Ergebnisse der SOA-Vision zu erhalten. Parallel zu den projektspezifischen und projektergänzenden Tätigkeiten wird die bereits erwähnte Hypothese, in Form eines Laborversuchs, verifiziert und validiert.

Die SOA beschreibt ein Konzept einer Software-Infrastruktur, in der die wesentlichen Funktionen einer Anwendung als Service realisiert sind. Diese Services werden in einer Registry zentral verwaltet. Somit können Services angeboten, gesucht und genutzt werden. Kennzeichnend sind hier Services mit entsprechender Granularität, die u. a. in neue Anwendungen integriert werden können. Ein hoher Integrationsgrad (durchgängige Unterstützung der Geschäftsprozesse) sowie eine hohe Integrationsbreite (Anzahl der integrierten Anwendungen ist hoch) zeichnen somit eine SOA aus. Dieses Konzept einer Software-Architektur verspricht mehr Agilität (in Bezug auf geplante sowie nicht geplante Veränderungen in der Zukunft) und (in seiner letzten Ausbaustufe) Wiederverwendung fachlicher Funktionalitäten.

Exemplarisch wird kurz die konkrete Projektanforderung eines SOA-Projektes skizziert: Durch das neue Architekturparadigma SOA soll, in dieser konkreten Projektanforderung, eine Hostwelt modernisiert und mittel bis langfristig eine weiche Migration durchgeführt werden. Bereits zur Jahrtausendwende wurde das Front-End, eine 3270-Emulation, durch eine Java-Applikation ersetzt. Zwei Jahre später wurde ein Applikations-Server eingeführt um das Front-End in Form eines Applets webfähig zu machen. Das aktuelle System ist eine klassi-

sche 3-Tier-Architektur. Das erste Tier, der Client, ist als Rich-Client in Java implementiert. Das zweite Tier, die Geschäftslogik, ist in COBOL auf dem Host realisiert. Das dritte Tier, die permanente Datenspeicherung, ist eine DB2 Instanz – ebenfalls auf dem Host. Die Notwendigkeit der weichen Migration wird durch mangelnde Verfügbarkeit von Experten, hohe Komplexität, steigende Wartungskosten, geringe Offenheit und mangelnde Anpassbarkeit begründet.

Das übergreifende Projekt koordiniert alle SOA-Aktivitäten und ergänzt diese u. a. mit einem Security-Konzept, um die Maschine-Maschine Kommunikation zu standardisieren. Weitere Zielsetzungen sind die Erarbeitung einer Enterprise Architektur für die MCG mit Betrachtung von SOA, Web Services und EAI-Konzepten; Ermittlung der Korrelationen zu den Prozessen; Betrachtung von Organisation und Governance; und die Verwendung strategisch positionierter Produkte.

Folgend wird der Ansatz zur Integration der Semantiken in einer SOA beschrieben. Dieser Ansatz wird im Labor basierend auf echten SOA-Projektanforderungen implementiert. Die Mächtigkeit und die Varianz der Propositionen (logische Aussagen) in OWL/OWL-S soll in Form eines Laborversuches aufgezeigt werden.

3. Der semantische Ansatz

Ziel ist das Erkennen äquivalenter Services in der Mercedes Car Group (DaimlerChrysler AG) durch logisches Schließen zu ermöglichen. Das Themengebiet der Logik ist in unterschiedlich mächtige Sprachen untergliedert. Bspw. befasst sich die Aussagenlogik mit Fakten und den zuordenbaren Werten wahr (1) oder falsch (0). Folglich hat eine Aussage Y zu einem bestimmten Zeitpunkt Z den Wert 1 oder 0. Ein Agent, kann den Wert einer Aussage zu einem bestimmten Zeitpunkt kennen oder nicht. Folglich kennt der Agent die Werte 1, 0 oder unbekannt. Die Logik erster Stufe befasst sich mit Fakten, Objekten und Relationen; zuordenbare Werte sind ebenfalls wahr (1) oder falsch (0). Die Fuzzy-Logik hingegen befasst sich mit Fakten in einem Wahrheitsgrad ($\in [0,1]$) innerhalb eines bekannten Intervallwertes. [1]

3.1. Beschreibungslogik (Description Logic)

Wissen kann in Form einer Ontologie repräsentiert werden. Ziel der Beschreibungslogik ist die Modellierung von domänenspezifischem Wissen (Ontologien) in einer wohldefinierten und somit eindeutigen Sprache. „Eine Ontologie ist eine formale Spezifikation einer gemeinsamen Konzeptualisierung“ [2]. Ziel ist es ein abstraktes Modell von spezifischen Aspekten der Realität zu erstellen. Generell wird zwischen A-Box (assertional box) und T-Box (terminological box) unterschieden. Die T-Box, ein Schlüsselelement der Beschreibungslogik, ist die Möglichkeit Terminologien zu bilden, um Konzepte zu beschreiben [3]. Im Folgenden wird das Konzept Frau aus der Schnittmenge Person und Weiblich definiert:

$$Frau \equiv Person \sqcap Weiblich$$

Der Bereich der A-Box ergänzt den Bereich der T-Box mit Wissen über die Klasseninstanzen: *Barbara* ist eine Instanz der Klasse *Frau* und hat eine Tochter. Um das Themengebiet in der Domäne Automotive zu verdeutlichen wird das Konzept eines Roadster in vereinfachter Form definiert: „Ein Roadster ist ein PKW mit genau 2 Türen und genau 2 Sitzen“. Modelliert man dieses Konzept mit den Mitteln der Beschreibungslogik so führt dies zu folgender Bedingung:

$$\begin{aligned} Roadster \equiv\ & PKW \sqcap \neg\ LKW \sqcap \\ & (= 2\ \text{hatTueren}) \sqcap \\ & (= 2\ \text{hatSitze}) \end{aligned}$$

3.2. OWL (Ontology Web Language)

Die Ontology Web Language (OWL) ist ein W3C Standard und basiert auf DAML-OIL. [4] Das Semantic Web [5] zeigt eine zukunftsweisende Vision auf, indem Informationen nicht mehr allgemein gültig, sondern explizit zuordenbar sind. Das primäre Ziel ist es Wissen maschinenlesbar zu modellieren, um durch einen automatisierten Prozess auf expliziertes Wissen zugreifen zu können. XML, XML Schema, RDF und RDF Schema sind für diese Art der Wissensrepräsentierung nicht ausreichend mächtig. OWL erfüllt durch sein erweitertes Vokabular mit der Möglichkeit disjunkte

Klassen zu kennzeichnen, transitive Eigenschaften zu definieren, etc. die Anforderung an eine formale Sprache. Drei standardisierte Ausprägungen (Lite, DL und Full) mit unterschiedlichen Ausprägungen sind verfügbar. Wir fokussieren unsere Aktivitäten auf OWL DL (Description Logic), da diese Ausprägung ausreichend mächtig und entscheidbar ist. [6]

OWL definition	**Paraphrase**
Class(Thing partial ...	All Things ...
Class(Thing complete parent...	A Thing is any Parent that ...
(add to all descriptions and definitions)	...amongst other things...
allValuesFrom	only
someValuesFrom	some
and	both... and also
not(... and ...)	not all of / not both ... and also
not (... or ...)	neither ... nor ...
someValuesFrom not	has some ... that are not ...
not (someValuesFrom ...)	does not have ... any
AllValuesFrom not	has ...no ... / has only ...that are not ...
not (allValuesFrom ...)	does not have ... only
subclassOf(A , B)	A implies B

Abbildung 2: Die Mächtigkeit von OWL [7].

Exemplarisch sollen mit OWL DL verfügbare Services aus einem Prozessteil semantisch modelliert werden. Ziel ist es ein allgemeingültiges Konzept für den Bereich Automotive (T-Box) entstehen zu lassen, der dann spezifisch für die Anforderungen von DaimlerChrysler erweitert (A-Box) wird.

3.3. OWL-S (OWL für Services)

OWL-S dient zur Beschreibung der Semantiken von Services, den sog. Semantic Web Services. Hier erfolgt bereits eine Festlegung auf die Technology Web Services. Ziel ist, basierend auf der semantischen Beschreibung, dynamisch zur Laufzeit Web Services zu suchen, aufzurufen und ggf. zu orchestrieren. [8]

OWL-S besteht aus 3 Bereichen: Service Profile, Service Grounding und Service Model. Das Service Grounding interagiert mit WSDL (Web Services Description Language) Dokumenten der Services und regelt die Ein- und Ausgabeparameter.

Das Service Model dient dazu den Datenfluss innerhalb des Services zu beschreiben. Es sind 3 verschiedene Typen von Prozessen verfügbar. Atomare Prozesse (atomic process), die direkt ausführbar sind und keine Unterprozesse beinhalten. Einfache Prozesse (simple process), die zum spezifizieren abstrakter Sichten dienen und nicht ausführbar sind. Kombinierbare Prozesse (composite process), die Kombinationen von atomaren, einfachen und kombinierbaren Prozessen ermöglichen [9; 10].

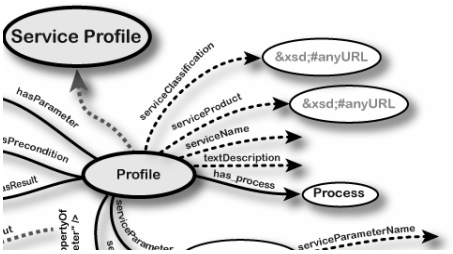

Abbildung 3: Service Profile (Ausschnitt) [11].

Das Service Profile spezifiziert die Eingaben, die ein Web Service benötigt; die Ausgaben, die er generiert; die Vorbedingungen, die erfüllt sein müssen und die Auswirkungen nach dem Ausführen (IOPE [12]). Das Service Profile und die Praxistauglichkeit von OWL DL liegen im Fokus unser Untersuchungen, Services semantisch zu modellieren. Bspw. ermöglicht der Tag <*serviceClassification*> im Service Profile das Verlinken auf Ontologien, die in OWL spezifiziert sind. Diese Kombination aus standardisierten Technologien soll exemplarisch den Zugriff auf semantisch modellierte Web Services zu Laborbedingungen ermöglichen.

3.4. Semantiken in SOA

Ziel der losen Kopplung ist, auf Änderungen in der Zukunft schneller reagieren zu können – Agilität. Client/Server wurden in den späten 80ern in Form von PC's populär. Diese 2-Tier-Architektur und die Software auf diesen Systemen verbesserte u. a. die Flexibilität/Agilität [13]. In den 90ern wurde ein weiterer Tier eingeführt – die Middleware. Primäres Ziel der Middleware ist, eine Kommunikation für ho-

mogene und heterogene Plattformen anzubieten [14]. Seit Mitte der 90er Jahre haben sich 3-Tier-Architekturen als Standard für Unternehmensanwendungen etabliert. Technologien wie CORBA [15] und DCOM [16] wurden durch die zurzeit populären Web Services ersetzt [17]. Der Bedarf Applikationen agiler zu gestalten als zuvor wächst mit der steigenden Anforderung an die Agilität der Unternehmen.

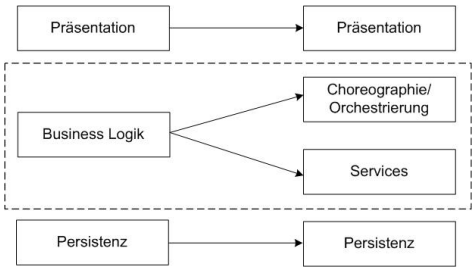

Abbildung 4: 4-Tier-Architektur.

Im Gegensatz zu den 4-Tier-Architekturen [18; 19], die das erste Tier (Präsentation) in zwei Tiers trennen; wird in diesem Ansatz das zweite Tier die Geschäftslogik (Business Logik) in „Komposition und Aggregation“ (Orchestrierung) sowie „Geschäftsregeln und Workflow“ (Choreographie) geteilt. Dadurch wird die klassische 3-Tier-Architektur zu einer 4-Tier-Architektur. Ziel ist es, Steuerung und Funktionalität voneinander zu trennen. Das zweite Tier (Choreographie & Orchestrierung) repräsentiert primär die Sicht des Fachbereichs und bildet die fachlichen Anforderungen in Form eines „Workflow“ ab. Das dritte Tier (Services) unterstützt die Bereitstellung der Funktionalitäten eines Systems in Form von „Building Blocks“[20]. Somit ist es möglich die Business Logik (dargestellt durch „Building Blocks“) in Services zu kapseln und die IOPE’s semantisch zu modellieren.

Um Services orchestrieren zu können, muss die Bedeutung dieser klar spezifiziert werden. Servicename sowie Ein- und Ausgabeparameter sind für eine zukunftsweisende Agilität und Wiederverwendung nicht ausreichend. Die Kernprozesse und unterstützende Prozesse im Unternehmen sowie die bereits gekapselten Services müssen spezifischer modelliert werden. Es wird mit der exemplarischen Modellierung mit dem aus echten Projektanforderungen zu realisierenden Geschäftsprozess begonnen. Ontologien werden erstellt um relevantes Unternehmenswissen abzubilden. Folgend wird exemplarisch ein Produkt aus fachlicher Sicht semantisch modelliert.

Notwendige und hinreichende Bedingungen sind nötig, um eine Klasse zu definieren. Fehlt die hinreichende Bedingung bei der Definition einer Klasse, so nennt man diese „primitive“ Klasse. Primitive Klassen können nicht für logisches Schließen (Inferenz) genutzt werden. Die Maschine interpretiert diese Form wie folgt: Die Klasse A hat, unter anderen Eigenschaften, die Eigenschaften a, b, c. Das folgende Beispiel visualisiert die definierte Klasse „Roadster“.

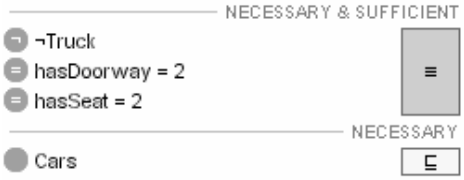

Abbildung 5: Die definierte Klasse Roadster.

```
<owl:Class rdf:ID="Roadster">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf
            rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty>
              <owl:FunctionalProperty
                  rdf:ID="hasDoorway"/>
            </owl:onProperty>
            <owl:cardinality rdf:datatype="…">
               2</owl:cardinality>
          </owl:Restriction>
          <owl:Restriction>
            <owl:cardinality rdf:datatype="…">
               2</owl:cardinality>
            <owl:onProperty>
              <owl:FunctionalProperty
                  rdf:ID="hasSeat"/>
            </owl:onProperty>
          </owl:Restriction>
          <owl:Class>
            <owl:complementOf>
              <owl:Class rdf:about="#Truck"/>
            </owl:complementOf>
          </owl:Class>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="#Cars"/>
</owl:Class>
```

Abbildung 6: Die codierte Klasse Roadster.

Semantic Web Services verfolgen zwei Ziele. Zum einem das dynamische Finden und Binden von Services zur Laufzeit und zum anderen das Orchestrieren eines spezifischen Services durch eine Kollektion von bereits zur Verfügung stehenden Services. Führt man dieses Gedankenexperiment fort, so werden in Zukunft Services weltweit zur Verfügung stehen, die aufgrund Ihrer semantischen Beschreibung gefunden werden. Nimmt man weiter an, dass Axiome in Form von Ontologien weltweit standardisiert zur Verfügung stehen, so kann dieses Ziel nur dann erfüllt werden, wenn uns formale Sprachen zur Verfügung stehen, die mächtig genug sind, um fachliche Anforderungen abzubilden und gleichzeitig den Raum für unscharfe Definitionen (im Sinne der Inferenz) minimieren.

3.5. Versuchsaufbau

Abgleitet von der, im Abschnitt Kurzfassung dargestellten, Hypothese werden folgende Thesen näher untersucht: Ist die Mächtigkeit der Technologie OWL ausreichend, um die fachlichen Anforderungen im Bereich Automotive formal abzubilden? Welchen Deckungsgrad, bezogen auf Äquivalenz, haben die modellierten Services basierend auf T-Box und A-Box? Welche Aussagen sind vom abweichenden Dekkungsgrad der Äquivalenten Services abzuleiten? Welche Restriktionen müssen erfüllt werden, um Services system- bzw. organisationsübergreifend modellieren zu können?

Eine Ontologie wird erstellt, die das Konzept PKW beschreibt und als Basis für die MCG-spezifische Ontologie dient. Diese spezifische Ontologie wird erstellt, um die Konzepte „Produkte“, „Aggregate“, „Einzelteile“ und „IT-Systeme“ des Beispielprozesses zu beschreiben. Zehn IT-unterstützende Services werden aus dem Beispielprozess selektiert und fachlich in einer eindeutigen Sprache [21] spezifiziert.

Der Laborversuch besteht aus zwei Versuchsreihen. In der ersten Versuchsreihe werden die IOPE’s der fachlichen Services, basierend auf dem Konzept PKW, modelliert. In der zweiten Versuchsreihe werden die IOPE’s der fachlichen Services, basierend auf der MCG-spezifischen Ontologie, modelliert. Das Laborjournal dient der Reproduzierbarkeit des Laborversuchs. Die o. g. Thesen werden durch den Laborversuch verifiziert und validiert.

4. Zusammenfassung und Ausblick

Das vorgestellte Projekt mit strategischer Ausrichtung soll Projekte, die sich mit Serviceorientierten Architekturen beschäftigen koordinieren und fehlende Bestandteile der SOA Vision operativ einbringen. Nach Abschluss des Projektes sollen konsolidierte Ergebnisse in Form von Best Practices, Guidelines, ein Konzept zur Serviceorientierten Organisation und ein Architektur-Blueprint zur Verfügung stehen.

Zusätzlich wird die SOA Vision konzeptuell durch die semantische Modellierung unter Laborverhältnissen ergänzt. Das semantische Beschreiben der Eingaben, Ausgaben, Vorbedingungen und Nachbedingungen in der Domäne Automotive im Labor, versprechen valide Messergebnisse zu liefern, um die Hypothese/Thesen zu verifizieren und zu validieren.

Diese Forschungsarbeit stellt somit, durch die Ergänzung des semantischen Ansatzes, eine sehr gute Ergänzung zur Aufgabenstellung des Projektes OrViA [22] dar.

Literatur

[1] *Russel, Stuart; Norvig, Peter:* Künstliche Intelligenz – Ein moderner Ansatz. Pearson Studium.

[2] *Borst, Pim; Akkermans, Hans; Top, Jan:* Engineering ontologies. In: International Journal of Human-Computer Studies 46 (1997) 2, S. 365–406.

[3] *Baader, Franz; Calvanese, Diego; McGuinness, D.; Nardi, Daniele; Patel-Schneider, Peter:* The Description Logic Handbook. Cambridge University Press, 2003.

[4] *Connolly, D.; Harmelen, F. v.; Horrocks, I.; McGuinness, D. L.; Patel-Schneider, P. F.; Stein, L. A.:* Reference Description W3C Note 18 December 2001. http://www.w3.org/TR/daml+oil-reference, Abruf am 2006-08-24.

[5] *Semantic Web:* Semantic Web Activity. http://www.w3.org/2001/sw/, Abruf am 2006-08-24.

[6] *McGuiness, D. L.; Patel-Schneier, P. F.:* OWL Web Ontology Language Overview. http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.2, Abruf am 2006-08-24.

[7] *Recto, Alan; Drummond, Nick:* OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns.

[8] *Roman, D.:* Web Services Modeling Ontology. http://www.wsmo.org, Abruf am 2006-08-24.

[9] *Bechhofer, S.; Harmelen, F. v.; Hendler, J.; Horrocks, I.; McGuinness, D. L.; Patel-Schneider, P. F.; Stein, L. A.:* OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/ #app-DLinRDF, Abruf am 2006-08-24.

[10] *Balzer, S.; Liebig, T.; Wagner, M.:* Pitfalls of OWL-S — A practical Semantic Web Use Case. 2nd International Conference on Service Oriented Computing ICSOC, 2004.

[11] *OWL-S:* Semantic Markup for Web Services. http://www.daml.org/services/ owl-s/1.1/overview/, Abruf am 2006-08-24.

[12] *Martin, David:* OWL-S: Semantic Markup for Web Services. SRI International. http://www.daml.org/services/ owl-s/1.1/overview/, Abruf am 2006-08-24.

[13] http://www.sei.cmu.edu/str/descriptions/ clientserver.html, Abruf am 2006-08-24.

[14] *Bernstein, Philip A.:* Middleware: A Model for Distributed Services. In: Communications of the ACM 39 (1996) 2, S. 86–97.

[15] Object Management Group home page. http://www.omg.org, Abruf am 2006-08-24.

[16] *Microsoft Corporation:* Distributed Component Object Model Protocol-DCOM/1.0. November 1996. http://www.microsoft.com/Com/ resources/comdocs.asp, Abruf am 2006-08-24.

[17] *Daskalova, Hristina:* Web Integration for Enterprise Applications. Institute of Information Technologies -BAS. Sofia, Bulgaria.

[18] *Saimi, Akihiro; Syomura, Tsutomu; Suganuma, Hiroshi; Ishida, Itaru:* Presentation Layer Framework of Web Application Systems with Server-Side Java Technology. The Twenty-Fourth Annual International Computer Software and Applications Conference, 2000.

[19] *Kozaczynski, W.:* Architecture Framework for Business Components. International Conference on Software Reuse ICSR'98, 1998.

[20] *Steiert, H-P.:* Towards a Component-based n-Tier C/S-Architecture. ISAW, Orlando Florida USA 1998.

[21] *Network Working Group; Bradner, S.:* Key words for use in RFCs to Indicate Requirement Levels. Harvard University, 1997. http://www.ietf.org/rfc/ rfc2119.txt, Abruf am 2006-08-24.

[22] *Stein, Sebastian; Kühne, Stefan; Wagner, Julia:* Das Forschungsprojekt OrViA – Orchestrierung und Validierung integrierter Anwendungssysteme. In: *Fähnrich, Klaus-Peter; Kühne, Stefan; Speck, Andreas; Wagner, Julia (Hrsg.):* Integration betrieblicher Informationssysteme, Problemanalysen und Lösungsansätze des Model-Driven Integration Engineering. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig 2006, S. 3–12.

[23] *McGuinness, Deborah L.; da Silva, Paulo Pinheiro:* Explaining Answers from the Semantic Web: The Inference Web Approach. Journal of Web Semantics, 2004. http://www.websemanticsjournal. org/ps/pub/2004-22, Abruf am 2006-08-24.