Matchmaking in Web Services

- Seminar Web Services
- Am 15.06.2004
- Von Joachim Jacobi
- und Stefan Graber

- Betreuer: Marcel Busse

Legende

- Definition
- Motivation / Anforderungen
- Suchsysteme
  - Keywordbasiert
  - Framebasiert
  - Ontologiebasiert
- Realisierung des Semantic Webs
  - RDF / DAML+OIL
  - DAML+S / DL
  - Racer
- Zusammenfassung / Ausblick

Definition: Was ist Matchmaking?

- Matchmaking is defined as a process that requires a repository host to take a query or advertisement as input, and to return all advertisements which may potentially satisfy the requirements specified in the input query or advertisement

- Matchmaking ist ein Prozess der eine gesuchte Menge A mit einer (sehr viel größeren) Menge B vergleicht und auf Überschneidungen prüft. Hierbei werden alle Teilmengen von B die A entsprechen als Treffer zurückgeliefert

Motivation: Wozu brauchen wir Matchmaking?

Beispiel:

- Wir wollen zur Bearbeitung unserer Diplomarbeit aus der Bibliothek passende Literatur heraussuchen.

Problem:

- Wir wissen nicht wo sich geeignete Literatur innerhalb der Bibliothek befindet.

Lösung:

- Einsatz eines geschickten Matchmakings

Anforderungen an das (webbasierte) Matchmaking aus der Praxis:

- Schnelles Finden von sinnvollen Items
  → zeitkritische Komponente

- Herausfiltern der unbenötigten Items
  → Problem der gleichen Syntax bei anderer Semantik

- Keine potentiell nützlichen Items vergessen
  → Problem der anderen Syntax bei gleicher Semantik

Matchmaking im Framework des W3C

UDDI:

- ist selbst ein Web Service, der als Verzeichnis zum Veröffentlichen und Auffinden von anderen Web Services fungiert
- “Gelbe Seiten für Dienste”

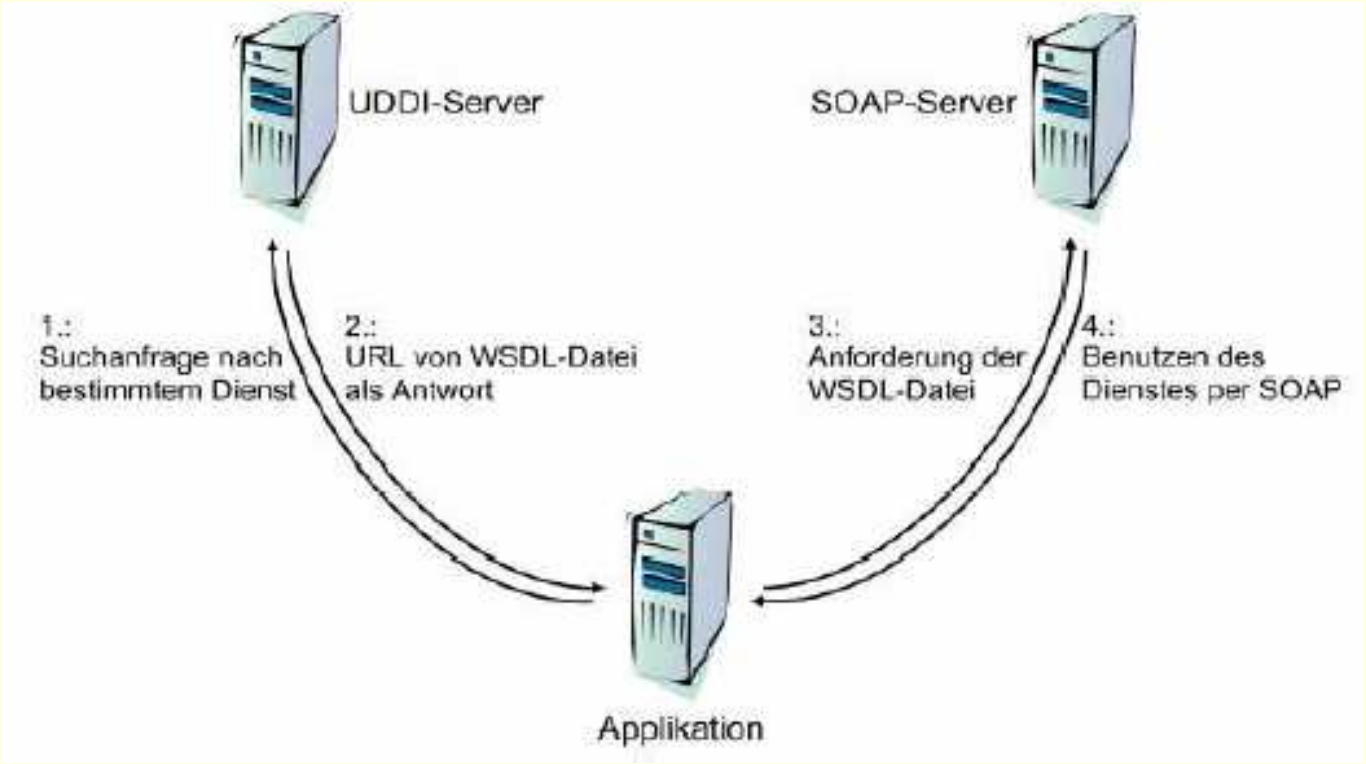
WSDL

- beschreibt die Operationen, welche Webservices ausführen
- “Aufruf eines Dienstes, wie sieht die Antwort aus?“

SOAP:

- “Umschlag für ein XML-Dokument”

Schematische Darstellung



- Das Matchmaking findet im UDDI Server statt!

Beispiel eines komplexen Web Services aus der Welt des e-Commerce

- Die „Bleicker Schrauben + Dübel GmbH“ bietet Schrauben zum Verkauf an
- Die Mindestbestellmenge liegt bei 5000 Stück
- Das Mindestgebot muss bei 20 Euro für 1000 Schrauben liegen
- Die Versandpauschale liegt bei 10 Euro
- Der Käufer muss über eine Mindestkreditwürdigkeit von 5 verfügen
- Die Schrauben werden bis spätestens 2 Tage nach Eingang der Bestellung ausgeliefert

  → Angebot wird auf dem UDDI Server platziert

Keywordbasiertes Suchsystem: Beispiel Google

- für jedes Schlüsselwort werden sämtliche Synonyme (Antonyme, Hyperonyme, Hyponyme) herausgesucht und in die Suche mit einbezogen

  → Motto: „Lieber mehr als zuwenig zurückliefern“

  → Für „html“ werden zum Bsp. die Begriffe „Script“, „Netscape“, „tutorial“ und „tag“ als Synonyme verwendet

- Google versucht die Qualität der Treffer zu erhöhen indem es prüft ob die Keywords auf verlinkten Seiten ebenfalls vorkommen

  → Trotz dieser Bemühung gibt es leider sehr viele falsche Hits

Keyword-basiertes Suchsystem:

Vorteile:

- Geignet bei der Suche nach einfachen Dateien oder Webseiten mittels Schlagworten
- Sehr schnelle Suche

Nachteile:

- Die Semantische Bedeutung kann nicht erfasst werden
  → Problem der gleichen Syntax bei anderer Semantik
  → Keine Einschätzung über die Relevanz der erhaltenen Treffer möglich

- Man muss wissen, wie das, was man sucht, heißt
  → Es reicht nicht zu wissen, was das, was man sucht, können soll
  → Anbieter und Kunde müssen sich über die Begrifflichkeiten einig sein!

→ Für die Suche nach Diensten nicht geeignet!

Framebasiertes Suchsystem

- Ein Frame besteht aus Attributen in denen die Eigenschaften eines Items beschrieben sind
  → Semantik wird aufgenommen
- Häufig kombiniert mit einem „Pre-Enumerated“ Vokabular von Service Arten und Eigenschaften
  → UDDI
- Query wird ebenfalls als Frame beschrieben
- Match kommt dann zustande, wenn die textuell beschriebenen Attribute zueinander passen

Framebasiertes Suchsystem: Beispiel

Beschreibung	Schraubenverkauf
Hersteller	Bleicker Schrauben + Dübel GmbH
MinBestellMenge	5000
PreisJe1000	20,00
Versandpauschale	10,00
MinKreditWürdigkeit	5
MaxZeitBisAuslieferung	48

- Unser Beispiel eines Web Services dargestellt durch einen Frame

Framebasierte Suche: Kritik

Vorteile:

- Es wird Semantik aufgenommen
- Suche nach Web Services ist möglich

Nachteile:

- Items und Querys müssen als Frame dargestellt werden
- Probleme in der Praxis?
- Wurde wirklich Semantik aufgenommen?

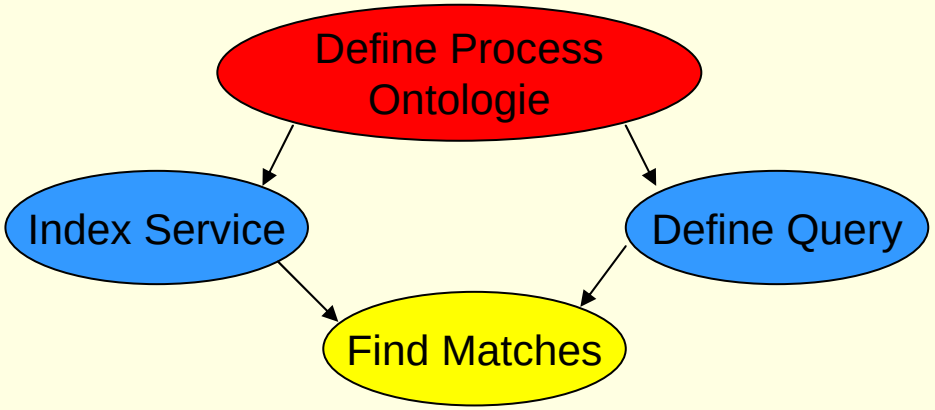
→ Besser als reine keywordbasierte Suche, aber geht es vielleicht noch besser?

Auf „Prozess Ontologien“ - basierte Suchsysteme

Definition Ontologie:

- Ein formal definiertes System von Dingen/Konzepten/Strukturen sowie dessen Beziehungen untereinander.
- In vielen Fällen handelt es sich bei den als Ontologien bezeichneten Strukturen lediglich um kontrollierte Vokabularien wie Klassifikationen oder Thesauri.
- Ontologien enthalten implizite Regeln
- Intuition: Eine Ontologie lässt sich vergleichen mit einer Datenbank
  → Struktur (Datenbankschema) und Inhalt (Daten) bilden ein Ganzes.

Schematische Darstellung dieser Architektur



- Alle Funktionen eines Services werden in einem Prozess-Modell festgehalten
- Anschließend wird es zum späteren wieder auffinden zusammen mit seinen Komponenten (subtasks) in die Ontologie integriert und indexiert.
- Die Anfragen sind ebenfalls als Prozess Modelle formuliert
- Der Matching Algorithmus findet nun all die Services, deren Prozess-Modells mit denen aus dem Query übereinstimmen.
- Der Vergleich findet anhand der semantischen Beziehungen, die in der Prozess Ontologie codiert sind, statt (später genauer)

Semantic Web: Ontologien für das Web

- The semantic web goal is to be a unifying system which will (like the web for human communication) be as un-restraining as possible so that the complexity of reality can be described

  (Tim Berners-Lee, the director of the World Wide Web consortium)

- Das grundlegende Konzept ist die adäquate Beschreibung der Informationen durch Metadaten und das Herstellen von semantischen Beziehungen zwischen Begriffen

  → Ontologien

- Es setzt auf dem gegenwärtigen Web auf und erweitert es um eine wohl definierte Bedeutung der Informationen

  → Ziel: Nicht nur maschinenlesbar, sondern Maschinen sollen auch „verstehen“ was sie da lesen

- Basiert zur Zeit auf XML und DAML+OIL als Sprachsyntax
- URLs dienen zur eindeutigen Identifizierung von Dokumenten bzw. Objekten.
- Auf diesem Konzept soll eine Vielzahl von Applikationen aufbauen.

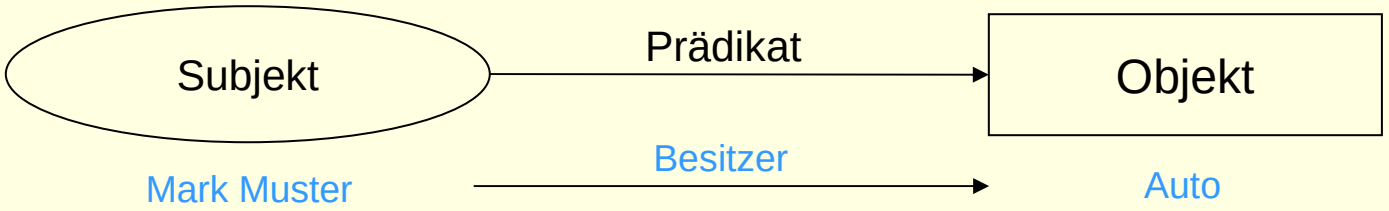
  → gutes Beispiel für eine Semantik Web Applikation ist das e-Commerce

RDF

RDF (Resource Description Framework): Modell zur Repräsentation von Metadaten (W3CStandard).

- RDF Syntax ist XML.
- RDF Model: Metadaten in Triple (Subjekt, Prädikat, Objekt) dargestellt

Bsp.: Max Muster ist der Eigentümer des roten Autos.
Der Eigentümer des roten Autos ist Mark Muster.



RDFS

RDFS (RDF Schema):

Erweiterung von RDF:

- Eigenschaften oder Beziehungen zwischen Eigenschaften
- Welche Ressource diese Eigenschaft besitzen darf
- Einfache Ausprägung von Werte- und Definitionsbereichen

RDFS fehlen wesentliche Bestandteile, um der Mächtigkeit von Ontologien gerecht zu werden (z.B. keine logischen Operatoren möglich)

DAML+OIL

Kurze Historik

DAML(DARPA Agent Markup Language)
Defense Advanced Research Projects Agency (DARPA), 1999
→ Es entstand auf Basis von RDFS die Ontologiesprache DAML-Ont.

OIL (Ontology Interchange Language)
OIL (ursprünglich Ontology Interference Layer)
→ europäisches Forschungsprojekt, gleiche Zielsetzung wie bei DAML.

Fusion zu DAML+OIL
DAML+OIL wurde als Standard für die Repräsentation von Metadaten und Ontologien im Internet beim W3C vorgeschlagen (2001).
→ W3C Arbeitsgruppe WebOnt zur Weitentwicklung von DAML+OIL gegründet.

DAML+OIL
Eigenschaften

- baut auf bestehenden Standards XML, RDF auf.
- objektorientiert
- Klassen, Subklassen, Eigenschaften, Subeigenschaften
- Äquivalenzausdruck um zu zeigen, dass zwei Instanzen einer Klasse oder einer Eigenschaft gleich sind
- maschinenlesbar und interpretierbar
- Mit logischen Operatoren können Axiome gebildet werden (Teilmenge, Vereinigungsmenge, Komplement

DAML-S

- Semantic markup languages wie DAML+OIL ermöglichen die Erstellung beliebiger Ontologien.
- DAML-S ist eine DAML+OIL Ontologie für Web Services.
- Das Ziel ist es, Web Services für Maschinen verständlich zu machen, um sie zur Lösung folgender Aufgaben zu befähigen:

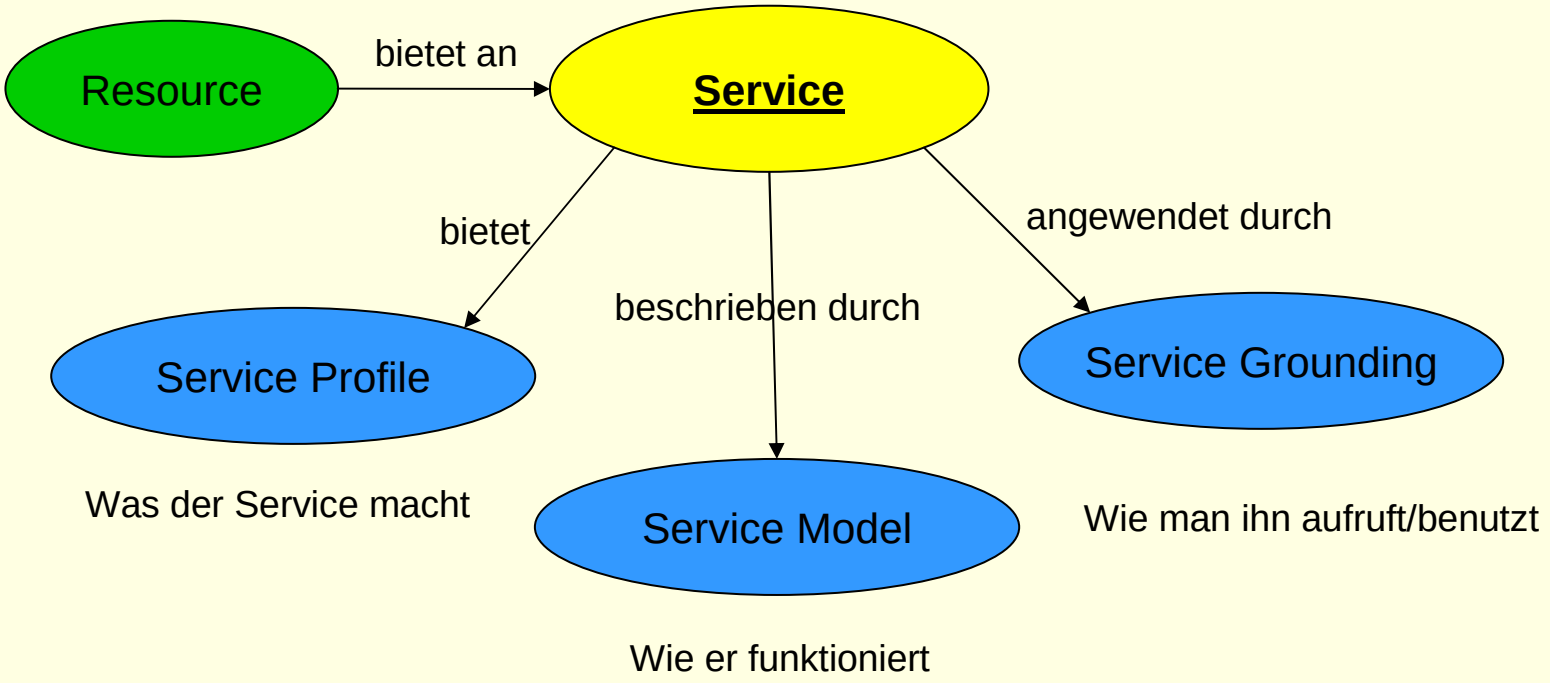
  → Suche/Entdeckung (UDDI)
  → Verifizierung der Service Eigenschaften
  → Aufruf eines Web Services durch einen anderen Service
  → Kontrolle bei der Ausführung
  → Zusammenarbeit

DAML-S

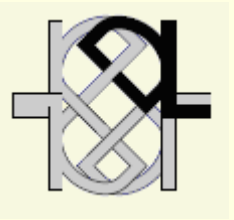
- Die DAML-S Ontologie bietet eine Menge von Klassen und Eigenschaften (Properties) an, um den Inhalt und die Fähigkeiten eines Services zu beschreiben.
- 3 Aspekte von DAML-S: Service Profile, Process Model und Service Grounding.
- Die Service-Ontologie beantwortet folgende Fragen:

    1. **Service Profile**:
       Was benötigt der Service von Agenten und was bietet er an?
    2. **Service Model**:
       Beschreibt den Arbeitsweise des Service.
    3. **Service Grounding**:
       Umschreibt detailliert die Art und Weise, wie der beschriebene Service mit anderen Services über Nachrichtenaustausch kommuniziert

DAML-S


Resource
bietet an
Service
bietet
beschrieben durch
angewendet durch
Service Profile
Service Model
Service Grounding
Was der Service macht
Wie er funktioniert
Wie man ihn aufruft/benutzt

Description Logic
Beispiel Ontologie



„Bleicker Schrauben + Dübel GmbH“ verkauft Schrauben über das Internet

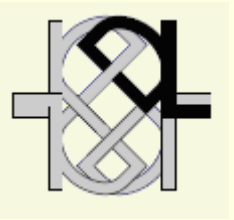
$$ServiceProfile \subseteq T$$
$$Adverstisement \subseteq ServiceProfile$$
$$Request \subseteq ServiceProfile$$

$$\begin{aligned} Sales \equiv\ & (=1\, providedBy.Actor), \\ & (=1\, requestedBy.Actor), \\ & (=1\, item.Equipment) \cap \\ & (=1\, hasQuantity.positiveInteger) \cap \\ & (=1\, canDeliver.Delivery) \end{aligned}$$

$$\begin{aligned} Delivery = & (=1\, location.DeliveryLocation), \\ & (=1\, date.DeliveryDate) \end{aligned}$$

$$\begin{aligned} Actor = & (=1\, hasName.ActorName), \\ & (=1\, hasCreditLevel.Integer) \end{aligned}$$

Description Logic
Beispiel Ontologie



Die Klasse Schrauben:

$$\begin{aligned} Schraube \subseteq\ & Equipment \cap \\ & (=1\, hasSort.Sort) \cap \\ & (=1\, hasSize.Size) \cap \\ & (=1\, hasPackagingSize.positiveInteger) \end{aligned}$$

$$Sort \equiv Holzschrauben \cup Blechschrauben \cup Sicherheitsschrauben$$

$$Size \equiv 1 \cup 2 \cup ... \cup 10$$

Inputs und Outputs:

$$inputs \subseteq parameter$$

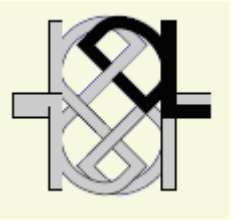
$$outputs \subseteq parameter$$

$$hasQuantity \subseteq inputs$$

$$hasUnitPrice \subseteq inputs$$

$$item \subseteq outputs$$

Description Logic
Beispiel Ontologie

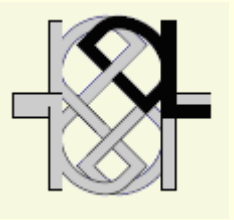


Beschränkungen im Advertisement (**Sales** und **Delivery)**

- Name: Bleicker S+D GmbH
- Schrauben
- Mindestens 5000 Stück pro Lieferung
- 20 € /1000 Stück
- Käufer hat eine Kredit-Würdigkeit von mindestens 5
- Lieferung innerhalb von zwei Tagen

$$
\begin{aligned}
Advert1 \equiv\ & (providedBy(Actor \cap \forall hasName.Bleicker),\\
& requestedBy(Actor \cap \geq_5 hasCreditLevel),\\
& profile(ServiceProfile \cap\\
& Sales \cap\\
& \forall item.(Schraube \cap \geq_{1000} PackagingSize) \cap\\
& \leq_{20} hasUnitPrice \cap\\
& \geq_{5000} hasQuantity \cap\\
& \forall delivery.(Delivery \cap\\
& \forall deliverydate.(\leq_2 afterOrder)))
\end{aligned}
$$

Description Logic
Beispiel Ontologie



Query (Anfrage)

- 10000 Blechschrauben der Größe 5 cm
- der Preis: weniger als 300 €, also weniger als 30/Packung

$$
\begin{aligned}
Request1 \equiv\ & profile(ServiceProfile \cap \\
& (Sales \cap \\
& \forall item.(Schraube \cap \forall hasSort.Blechschraube \cap \\
& =_5\ hasSize \cap \\
& \leq_{30}\ hasUnitPrice)))
\end{aligned}
$$

Matching Definition

Matchmaking wird hier als ein Prozess definiert, der einen Host benötigt, dort ein Query oder Advertisement eingibt und als Rückgabe alle Advertisements ausgibt, die möglicherweise den Anforderungen entsprechen, die in der Eingabe beschrieben werden.

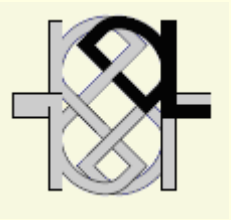
Formal: Sei α die Menge aller gespeicherten Services/Advertisements.
Für eine Eingabe Q (Query/Adv.) gibt der Matchmaking Algorithmus des Hosts die Menge aller kompatiblen Advertisments zurück, *matches(Q):*

$$matches\,(Q) = \left\{A \in \alpha \middle| compatible\,(A, Q)\right\}$$

Zwei Beschreibungen (Strukturen) sind kompatibel, wenn die Schnittmenge beider Strukturen nicht leer ist:

$$compatible(D1, D2) \Leftrightarrow \neg(D1 \cap D2 \subseteq \bot)$$

Description Logic Matching Definition



$$Request1 \equiv profile(ServiceProfile \cap (Sales \cap \forall item.(Schraube \cap \forall hasSort.Blechschraube \cap =_5 hasSize \cap \leq_{30} hasUnitPrice)))$$

$$Advert1 \equiv (providedBy(Actor \cap \forall hasName.Bleicker), requestedBy(Actor \cap \geq_5 hasCreditLevel), profile(ServiceProfile \cap Sales \cap \forall item.(Schraube \cap \geq_{1000} PackagingSize) \cap \leq_{20} hasUnitPrice \cap \geq_{5000} hasQuantity \cap \forall delivery.(Delivery \cap \forall deliverydate.(\leq_2 afterOrder))))$$

- Die Schnittmenge von Request1 mit Advert1 ist ausreichend, formal:

$$Advert1 \in matches(Request1)$$

Racer DL reasoner

- Was ist RACER?
- RACER ist eine Semantic Web Inference-Maschine zur

  → Erstellung von Ontologien
  → Query - Bearbeitung über RDF Dokumente und in Bezug auf RDFS/DAML Ontologien

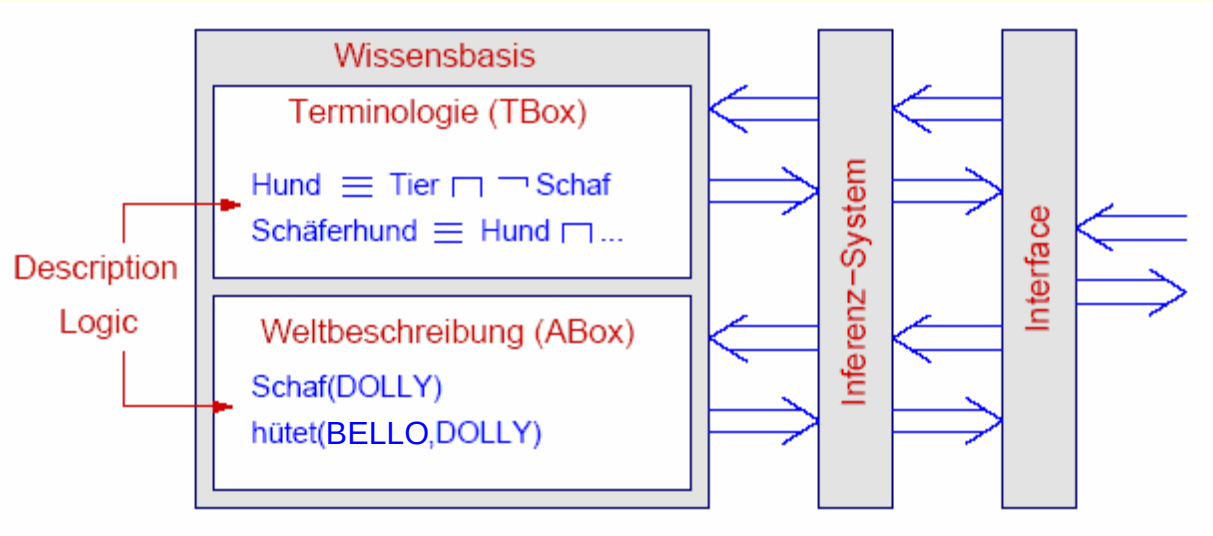
- RACER ist ein “Description Logic Reasoning System” in Verwendung mit

  → TBoxes
  → ABoxes

TBox/ABox

TBox: → intensionales Wissen (Terminologien bzw. Taxonomien)
→ zeitlich konstantes "Allgemeinwissen"
→ Definition von Klassen (Arbeiten mit Mengen)

ABox: → extensionales Wissen (assertional knowledge)
→ spezielles situationsabhängiges Wissen, das u.U. ständigen Änderungen unterworfen ist
→ Objekte (Instanzierungen der in der TBox definierten Klassen)



Matchmaking Algorithmus

Racer
Semantic Middleware for Industrial Projects Based on RDF/OWL, a W3C Standard

Prozess des Matchings einer Anfrage (Request):

1. Berechnet eine ServiceProfile Hierarchie für alle angebotenen Services.
2. Empfängt der Racer ein Request, stuft er das ServiceProfile ein.
3. Errechnet die Gemeinsamkeiten zu allen vorhandenen Advertisements.
4. Dann ordnet er sie nach den Level der Matches.

Verschiedene Level von Matches:

Exact :	Request R und Adv. A haben das gleiche Konzept, formal:	$A \equiv R$
PlugIn:	Request R ist ein Subkonzept Adv. A, formal:	$R \subseteq A$
Subsume:	R ist ein Superkonzept von A, formal:	$A \subseteq R$
Intersection:	Die Schnittmenge von A und R ist ausreichend, formal:	$\neg(A \cap R \subseteq \bot)$
Disjoint:	sonst formal:	$A \cap R \subseteq \bot$

Exact > PlugIn > Subsume > Intersection >Disjoint

Zusammenfassung / Ausblick

Keywordbasierte Suchsysteme:

- Leicht zu bedienen, vor allem für „Normalsterbliche“
- Spielt bei Web Services keine Rolle

Framebasierte Suche:

- Die aktuelle Version von UDDI basiert darauf
- Bedient sich letztlich der etablierten keywordbasierten Suche
- Wird in einigen Jahren durch das „Semantic Web“ abgelöst werden

Auf Prozess-Ontologien basierte Suche:

- Wird noch kaum eingesetzt, erste Realisationen vorhanden
- Durchbruch des „Semantic Webs“ kommt in den nächsten Jahren
- OWL