The Relation between Ontologies and Schema-languages:
Translating OIL-specifications in XML-Schema

Michael Klein, Dieter Fensel,
Frank van Harmelen and Ian Horrocks
(2000)

0.Vorschau

1. Einleitung
2. XML-Schema
3. OIL
4. Vergleich XML-Schema & OIL
5. Übersetzung : OIL in XML-Schema
6. Fazit

1. Einleitung

Semantic Web

„*The Semantic Web is an extension of the current web in wich information is given well-defined meaning, better enabling computers and people to work in cooperation.*“ (Tim Berners-Lee)

Ontologie

- wohldefiniertem Vokabular einer Domäne
- hierarchische Beschreibung von Konzepten (z.B. Student)
- Definition von Beziehungen zwischen Konzepten (z.B. Professor prüft Student)

1.Einleitung

Das World Wide Web:

- Große Menge an Daten, Organisationen, Communities, etc.
- Einfacher Zugriff (URLs, Links, Suchmaschinen)
- Standards :

  1.physikalische Ebene (HTTP, TCP, etc.)

  2.syntaktische Ebene (HTML)

➔ Erfolg, aber auch Probleme

1.Einleitung

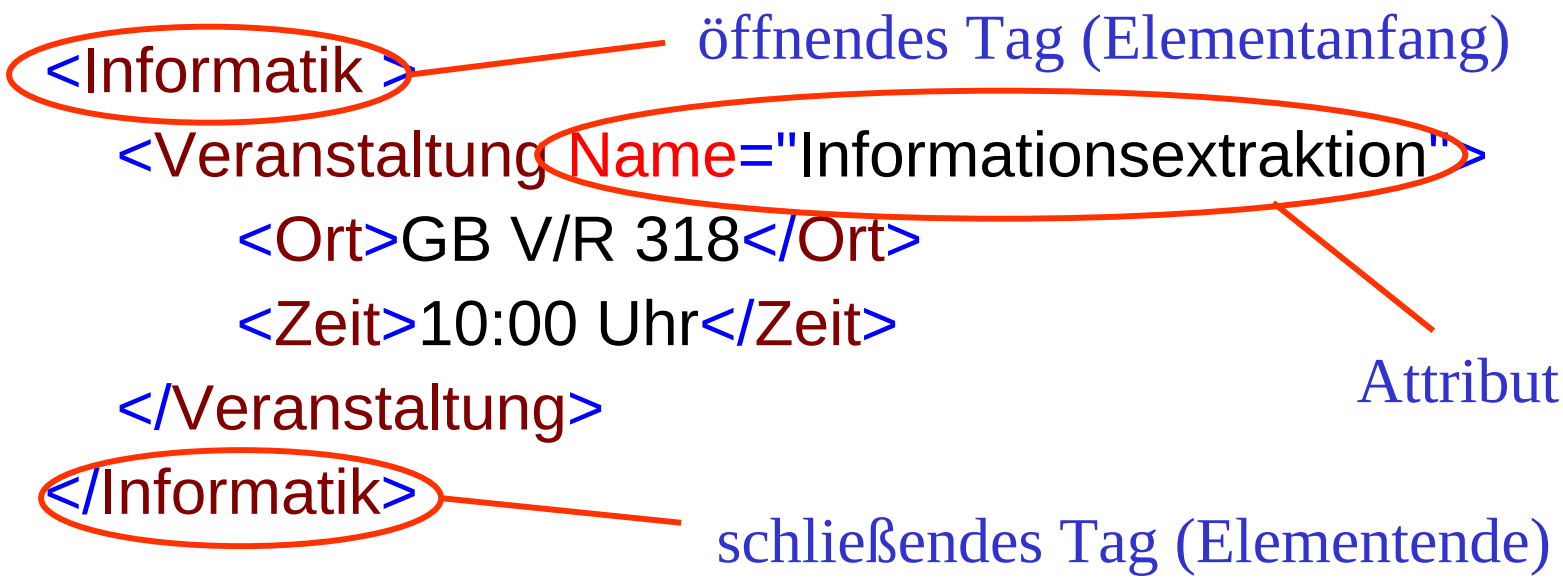
Problem : HTML (HyperText Markup Language)

- vermischt Layout- und Logische-Ebene
- geringe Anzahl logischer Tags (i.w. Titel, Paragraph, Aufzählungen)

  ➔ eingeschränkte logische Strukturierung
- geringe Anzahl layout Tags

➔ Lösung : **XML** (eXtensible Markup Language)

1.2 XML



- serielle Syntax für hierarchische Strukturen (Baumstruktur)
- beliebige Schachtelungstiefe
- Vergabe eigener Tag-Namen

2. XML-Schema

- Möglichkeit zur Einschränkung der Syntax und Struktur eines XML Dokumentes (wie DTD)
- Definition einer Klasse von XML Dokumenten
- **Vorteile zur DTD**:
  - XML-Schema ist selbst XML Dokument
  - Verfügt über große Anzahl Datentypen
  - verbesserte Möglichkeiten eingebettete Tags zu strukturieren
  - Verfügt über Mechanismus für Namensräume

2. XML-Schema

Beispiel:

```
<xs:element name=„Vorlesung">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Ort"  type="xs:string"/>
            <xs:element name="Zeit"  type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="Name" use="required"/>
    </xs:complexType>
</xs:element>
```

2.2 Datentypen

Wertebereich (value space):

- axiomatisch, per Aufzählung oder durch bilden von Teilmengen bereits definierter Datentypen

lexikalischer Bereich (lexical space):

- Wert eines Datentypen, kann durch ein oder mehrere Literale dargestellt werden (z.B. „100“ oder „1.0E2“)

Facetten:

- fundametale: Ordnung, obere/untere Grenzen, endlich/unendlich, etc.
- einschränkende: optional; schränken den ursprünglichen Bereich ein (max. Länge/Grösse, reguläre Ausdrücke)

Facettierung (1)

```
<xs:element name="Matrikelnummer">
    <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="10000"/>
            <xs:maxInclusive value="9999999"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

```
<Matrikelnummer>89372</Matrikelnummer>
```

Facettierung (2)

(eingebettete/lokale Definition)

```
<xs:element name=„Zeit">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:pattern value="[0-9]{2}:[0-9]{2} Uhr"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

```
<Zeit>10:00 Uhr</Zeit>
```

Facettierung (3)

(globale Definition)

```
<xs:element name="Zeit" type="t_Uhrzeit"/>

<xs:simpleType name="t_Uhrzeit">
    <xs:restriction base="xs:string">
        <xs:pattern value="[0-9]{2}:[0-9]{2} Uhr"/>
    </xs:restriction>
</xs:simpleType>
```

2.3 Strukturen

- Einschränkung des Inhaltes von Elementen und Attributen (s.o.)
- Vergrössern der Information von Instanzen, mittels default Werten und Typinformationen

Elemente:

- „einfache“ Elemente haben Datentypen als Inhalt
- komplexe Elemente haben weitere Elemente als Inhalt
- gemischte Elemente können beides beinhalten
- Zusätzlich können Elemente Attribute haben

XML-Schema

„Vererbung“ (derived type definition)

(Ableitung durch Erweiterung)

```
<xs:complexType name="Person">
    <xs:sequence>
        <xs:element name="Name"/>
        <xs:element name="Wohnort"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="t_Student">
    <xs:complexContent>
        <xs:extension base="Person">
            <xs:sequence>
                <xs:element name="Matrikelnummer">
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

derived type definition

- Ableitung eines Element-Typs mittels Erweiterung oder Einschränkung eines bestehenden

- Bildung von Klassenhierarchien nur explizit möglich!

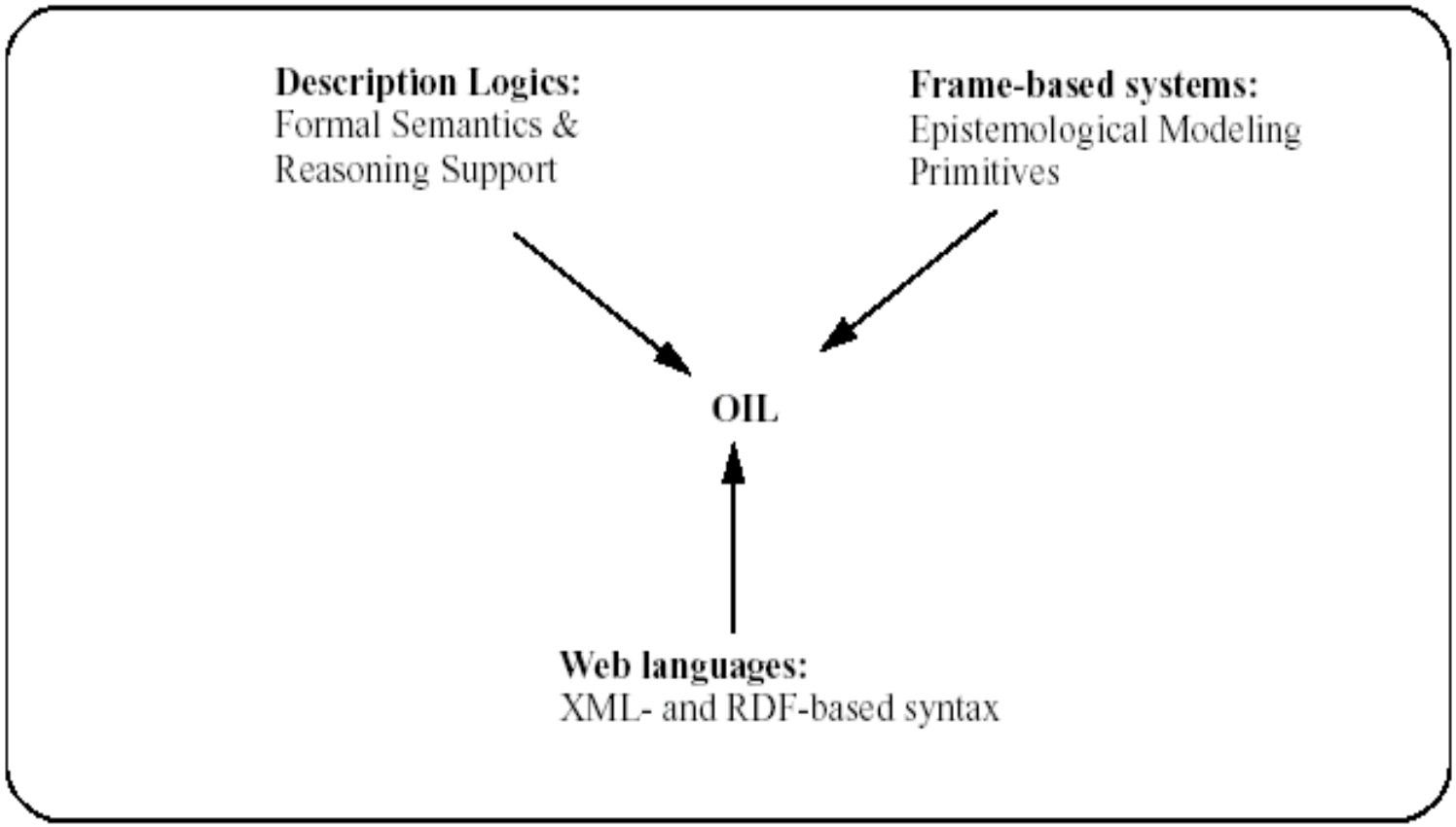
3. OIL

Ontology Inference Layer

- abstrakte Sprache zur Beschreibung von Ontologien
- modellierende Primitive aus Frame-basierten Ontologien
- wohldefinierte Semantik, auf Beschreibungslogik beruhend
- unterstützt automatisierte Schlussfolgerungen (wie Konsistenzprüfungen oder Klassensubsumtion)

3. OIL

Die drei „Wurzeln“ :



3.1 OIL Bestandteile

1. **Ontologie Behälter**
   - Enthält Meta-Informationen zur Ontologie selbst
   - Metadaten Elemente der Dublin Core Elementen Menge (title, creator, subject, etc.)

2. **Ontologie Definition**
   - Definition der Klassen (Konzepte) und ihren Beziehungen (Attribute bzw. Rollen)

3.2 Ontologie Definition

Klassen-Definitionen (class-def):

- **name** : Name der Klasse
- **subclass-of**[?] :
  - muss Unterklasse aller class-expressions der Liste sein
- **slot-constraint**[*] :
  - Klasse muss Unterklasse aller slot-constraints sein
- **type**[?] : primitive oder defined
  - primitive = notwendige Bedingung
  - defined = hinreichende Bedingung

class-expression = class-name, slot-constraint oder boolesche Kombination mittels AND,OR oder NOT

OIL

3.2 Ontologie Definition

class-def animal

class-def plant
    subclass-of NOT animal

class-def tree
    subclass-of plant

class-def branch
    slot-constraint *is-part-of*
        has-value tree

class-def defined carnivore
    subclass-of animal
    slot-constraint *eats*
        value-type animal

OIL

3.2 Ontologie Definition

Attribut-Definition (slot-def):

- **name** : Name des Slots
- **subslot-of**[?] **: slot-def** *daughter-of*
  subslot-of *child-of*
- **inverse**[?] **: slot-def** *eats*
  inverse *eaten-by*
- **properties**[?] : transitive, symmetric
  - transitive : **slot-def** *bigger-than*
    properties *transitive*

OIL

3.3 Ontologie Beispiel

Bemerkungen :

- Die Klassen **plant** und **animal** sind disjunkt
- Die Klasse **lion** ist Unterklasse von **carnivor** aufgrund ihrer Definition (da **carnivor** defined ist!)
- Die Klassen **herbivore** und **carnivore** sind disjunkt (OIL unterstützt mehrfaches Erben)
- Die Klasse tasty-plant ist inkonsistent

OIL

4. XML & OIL

Verschiedene Aufgaben :

- XML-Schemata bieten Strukturen zur Beschreibung von Informationsquellen (Dokumente oder semi-strukturierte Texte)
- OIL spezifiziert eine Domänen-Theorie (Ontologie)

Gemeinsames Ziel :

- Beide stellen ein Vokabular und Strukturen, die Informationsquellen beschreiben, bereit welche auf den Austausch ausgerichtet sind.

4.1 Vergleich

- Beide haben eine XML-Syntax
- XML-Schema hat vielfältige Datentypen, Oil nicht
- XML-Schema bietet eine Grammatik zur Ableitung von Elementstrukturen (sequence, choice), Oil nicht
- Vererbung ist in beiden vorhanden, allerdings:
  - XML-Schema wesentlich eingeschränkt

5. Oil in XML-Schema

Typdefinition :

```
<complexType name="carnivoreType" base="animalType" derivedBy="extension" content="mixed">
    <element name="eats" minOccurs="0">
        <complexType base="eatsType" derivedBy="extension">
            <element ref="animal" />
        </complexType>
    </element>
</complexType>

<complexType name="herbivoreType" base="animalType" derivedBy="extension" content="mixed">
    <element name="eats" minOccurs="0">
        <complexType base="eatsType" derivedBy="extension">
            <element ref="plant-or-is-part-of(plant)" />
        </complexType>
    </element>
</complexType>

<complexType name="giraffeType" base="herbivoreType" derivedBy="restriction" content="mixed">
    <element name="eats" minOccurs="0">
        <complexType base="eatsType" derivedBy="extension">
            <element ref="leaf" />
        </complexType>
    </element>
</complexType>
```

Übersetzung

5. Oil in XML-Schema

Elementdefinition :

```
<element name="carnivore" type="carnivoreType"/>
<element name="herbivore" type="herbivoreType"/>
<element name="eats" type="eatsType"/>
<element name="giraffe" type="giraffeType"/>
<element name="leaf" type="leafType"/>
```

Instanz :

```
<giraffe>
    <name>Iwan</name>
    <eats>
        <leaf>acacia-tree leaves</leaf>
    </eats>
</giraffe>
<lion>
    <name>Simba</name>
    <eats>
        <herbivore xsi:type="cowType">
            <type>Dikbil</type>
            <origin>Holland</origin>
        </herbivore>
    </eats>
</lion>
```

Übersetzung

5. Schlussfolgerung

XML-Schema und OIL dienen beide der Erweiterung von (online) Informationsquellen um eine semantische Schicht. Allerdings arbeiten beide auf unterschiedlichem Level der Abstraktion. Sie werden in unterschiedlichen Phasen der semantischen Beschreibung eingesetzt.

OIL dient der Modellierung einer Ontologie und XML-Schema dient als Syntax und strukturellen Definition der Quellen.

Übersetzung

6. Fazit

Fazit :

- Zum automatisieren von Datenaustausch und zur Vereinfachung von Informationsextraktion sind XML-Schema und OIL (in Kombination) wesentliche Erleichterungen
- Problem : Um die Vision eines „semantic Web“ zu realisieren muss beides auch Standard werden!