

Vorlesung Netzbasierte Informationssysteme

Semantic Web III

Prof. Dr. Adrian Paschke

Arbeitsgruppe Corporate Semantic Web (AG-CSW)

Institut für Informatik, Freie Universität Berlin

paschke@inf.fu-berlin.de

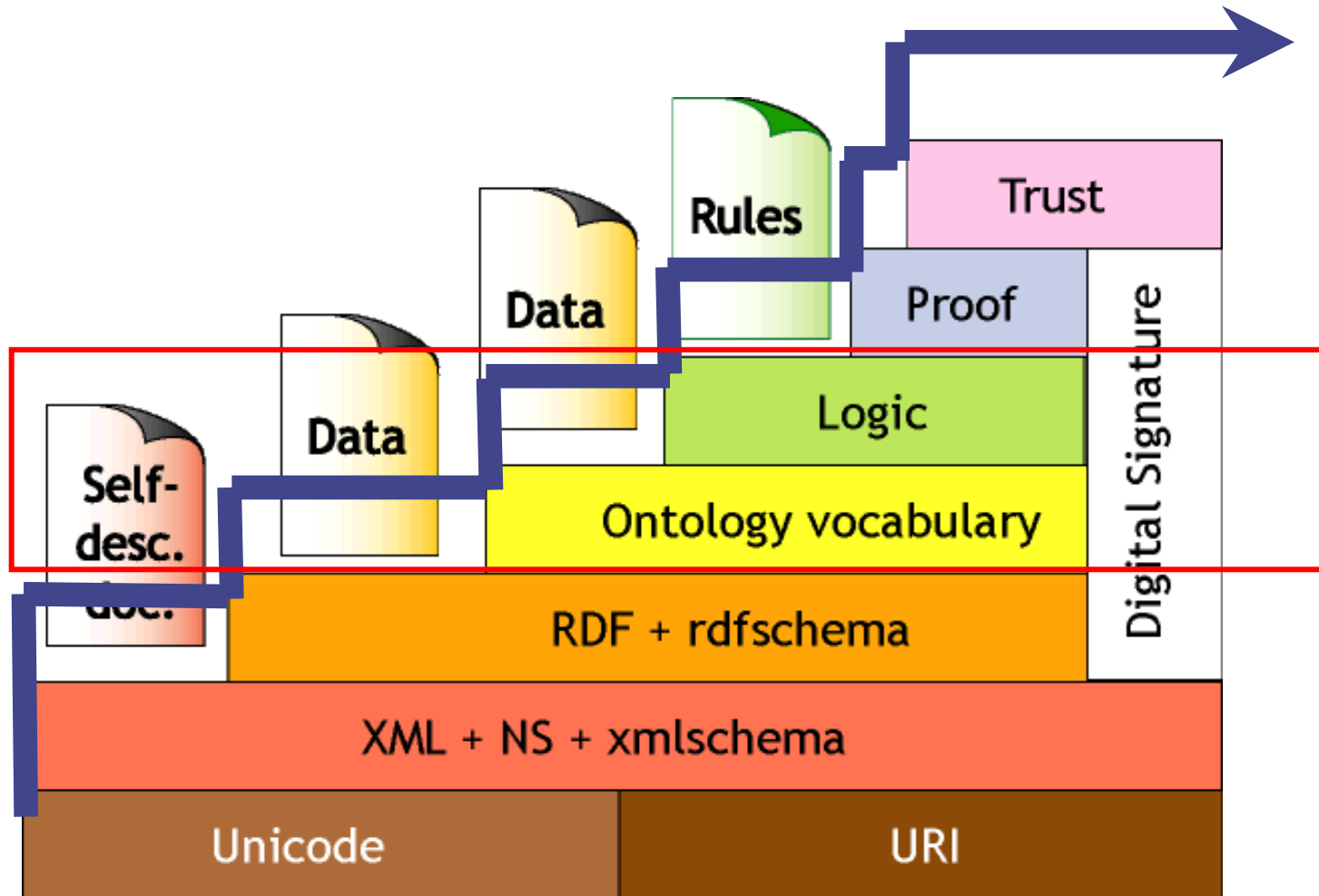
<http://www.inf.fu-berlin.de/groups/ag-csw/>



Semantic Web

- Semantic Web – Ein Einführung (Teil 1)
 - Semantic Web Vision
 - RDF
 - RDFS
 - RDF Anfragesprachen / SPARQL
- Semantic Web – Ontologie (Teil 2)
 - Was ist eine Ontologie
 - W3C Web Ontology Language (OWL)
 - OWL 1.1 und OWL 2
- Semantic Web – Regeln und Erweiterte Konzepte (Teil 3)

Semantic Web



Web Rules

Deklarative Programmierung mittels Regeltechnologie

*Users employ rules to express **what** they want, the responsibility to interpret this and to decide on **how** to do it is delegated to an interpreter*

Represent knowledge in a way that is understandable by 'the business', but also executable by rule engines, thus bridging the gap between business and technology

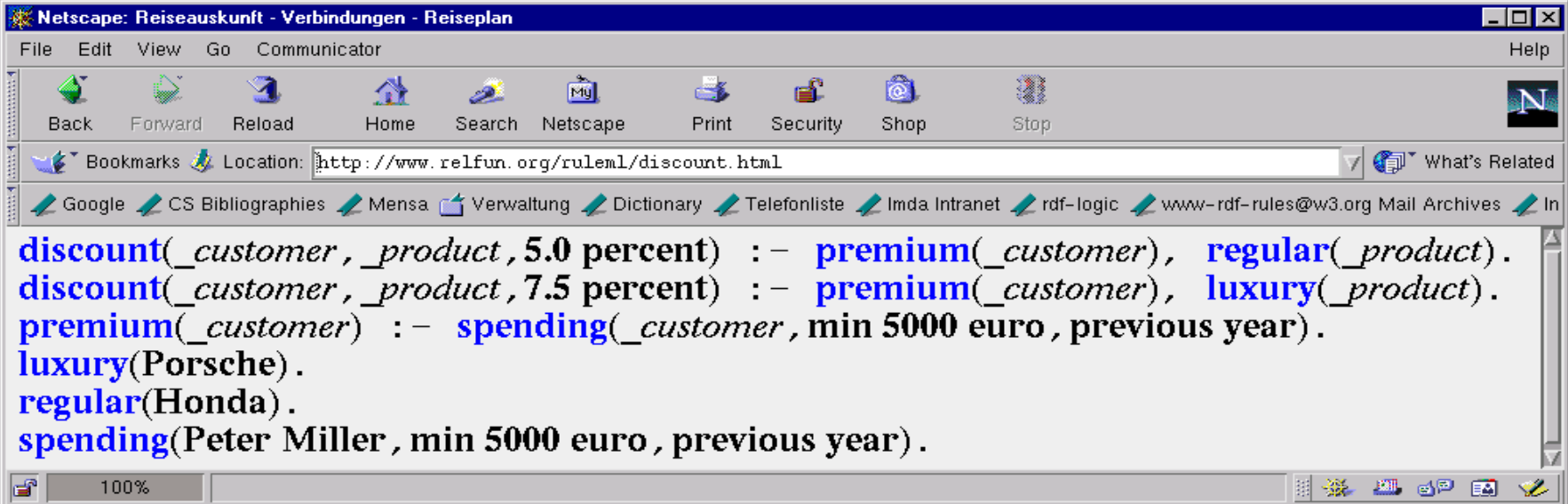
Beispiel: Deduktionsregeln im Web

Business Rules:

"The **discount** for a *customer* buying a *product* is **5.0 percent** if the *customer* is **premium** and the *product* is **regular**."

"The **discount** for a *customer* buying a *product* is **7.5 percent** if the *customer* is **premium** and the *product* is **luxury**."

Prolog-artige Formalisierung:



The screenshot shows a Netscape browser window titled "Netscape: Reiseauskunft - Verbindungen - Reiseplan". The address bar contains the URL "http://www.relfun.org/ruleml/discount.html". The main content area displays the following Prolog-like formalization:

```
discount(_customer, _product, 5.0 percent) :- premium(_customer), regular(_product).
discount(_customer, _product, 7.5 percent) :- premium(_customer), luxury(_product).
premium(_customer) :- spending(_customer, min 5000 euro, previous year).
luxury(Porsche).
regular(Honda).
spending(Peter Miller, min 5000 euro, previous year).
```

Exkurs: Logische Programmierung

- Logischer Programmierung
 - Logische Programme
 - Menge von Regeln und Fakten
 - Automatisierte Ausführung über effiziente Inferenzmaschinen (Regelmaschine)
 - Eine Inferenzmaschine versucht Anfragen an Hand der Fakten und Regeln zu *beweisen*.
 - Logische Programme als eingeschränkte Prädikatenlogik
 - Eine Klausel heißt **Horn-Klausel**, wenn sie **höchstens ein** positives Literal enthält:
 $\{\neg P_1, \neg P_2, \dots, \neg P_n, C\}$ entspricht $(P_1 \ P_2 \ \dots \ P_n) \rightarrow C$
 - Ableitungsregel (Derivation Rule):
 $Body \rightarrow Head$ (If Body then Head)
 $P_1 \ \dots \ P_n \rightarrow C$
 - Beispiel (Prolog Notation)
 $p0(X) :- p1(X), p2(a,b), p3(p4(X)).$

Beispiel

- Bonus-Malus Regelung

Dienstqualität (QoS)	Durchschnittliche Erreichbarkeit	Bonus/Malus Rabatt
Hoch	100 %	+ 5%
Normal	98-100 %	+ 0%
Niedrig	<98 %	- 5%
Unterdurchschnittlich	<95 %	1000 € Strafe

Beispiel: Formalisierung in Regeln

Wenn die durchschnittliche Erreichbarkeit = 100 %, dann ist die Dienstqualität hoch.

Bedingungen		
Prädikat	Komplexer Term	Konstante
=	availability(Service)	100%



Schlussfolgerung		
Prädikat	Variable	Konstante
qos	Service	high

Wenn die Dienstqualität hoch ist, dann wird ein Bonus von +5% auf den Basispreis berechnet.

Bedingungen		
Prädikat	Variable	Konstante
qos	Service	high

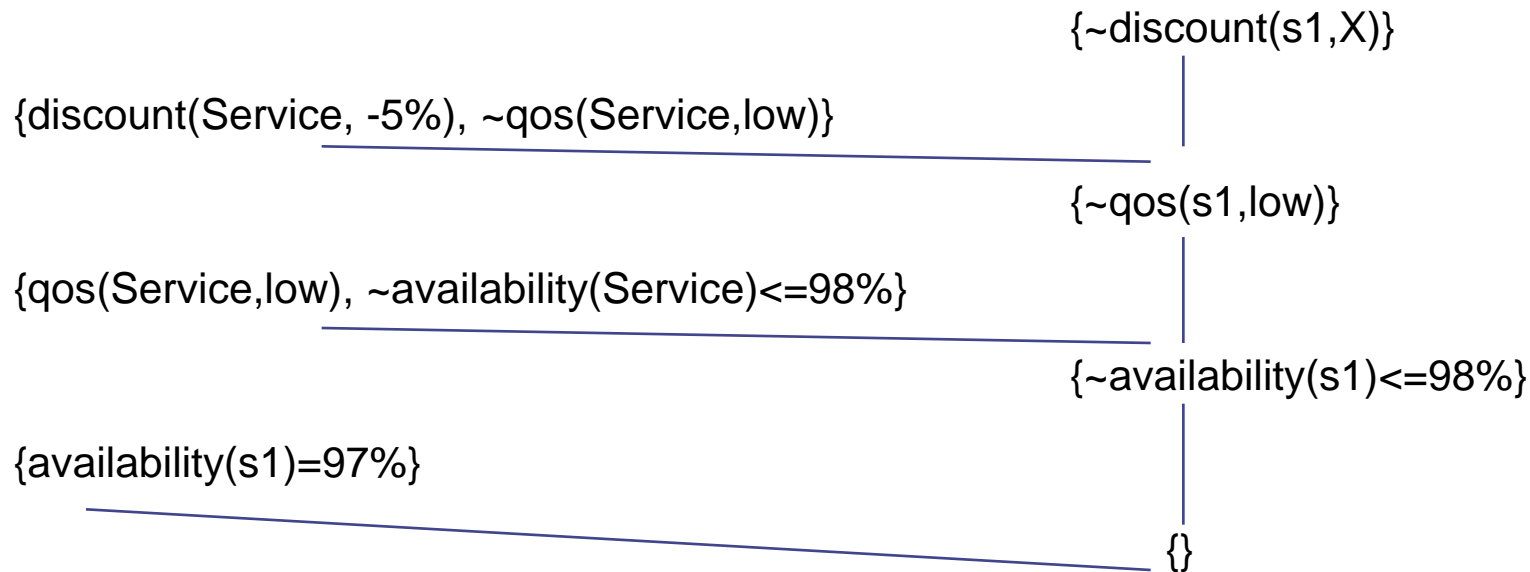


Schlussfolgerung		
Prädikat	Variable	Constant
discount	Service	5%



Inferenzmaschinen und Resolution

- Deduktive „Berechnung“ von Werten mittels (rückwärts-verkettetem) **Resolutionsalgorithmus**, die eingesetzt für die (existenzquantifizierten) Variablen, die Aussage (Anfrage) wahr machen.
- Beispiel:
 - Rules: $\text{discount}(\text{Service}, -5\%) \text{ :- qos}(\text{Service}, \text{low})$.
 - $\text{qos}(\text{Service}, \text{low}) \text{ :- availability}(\text{Service}, \text{ServiceAvailability}), \text{lessequ}(\text{ServiceAvailability}, 98\%)$.
 - Fakt: $\text{availability}(s1)=97\%$
 - Anfrage: $\text{discount}(s1, X)? \rightarrow$ Antwort: $X = -5\%$



Kompakte deklarative Wissensrepräsentation

Logische Programmierung

```
discount(Service, 5%) :- qos(Service,high).
discount(Service, -5%) :- qos(Service,low).
qos(Service,high):- availability(Service) = 1.
qos(Service,low):- availability(Service) < 0,98.
```

Anfragen

discount(Service,X)?	Alle Rabatte für alle Dienste
discount(s1,X)?	Rabatt für den Dienst „s1“
discount(s1,5%)?	Dienst „s1“ → Rabatt von 5%?
discount(Service,5%)?	Alle Dienste mit Rabatt von 5%
qos(Service,Y)?	Service Level aller Dienste?
qos(s1,Y)?	Service Level für Dienst „s1“?
...	

Imperative Programmierung

```
boolean getsDiscount(Service s, int value) {
    if (getAvailability(s)==1) && (value==1) return true;
    else if (getAvailability(s)<0,98) && (value<0,98) return true;
    else return false;
}

...

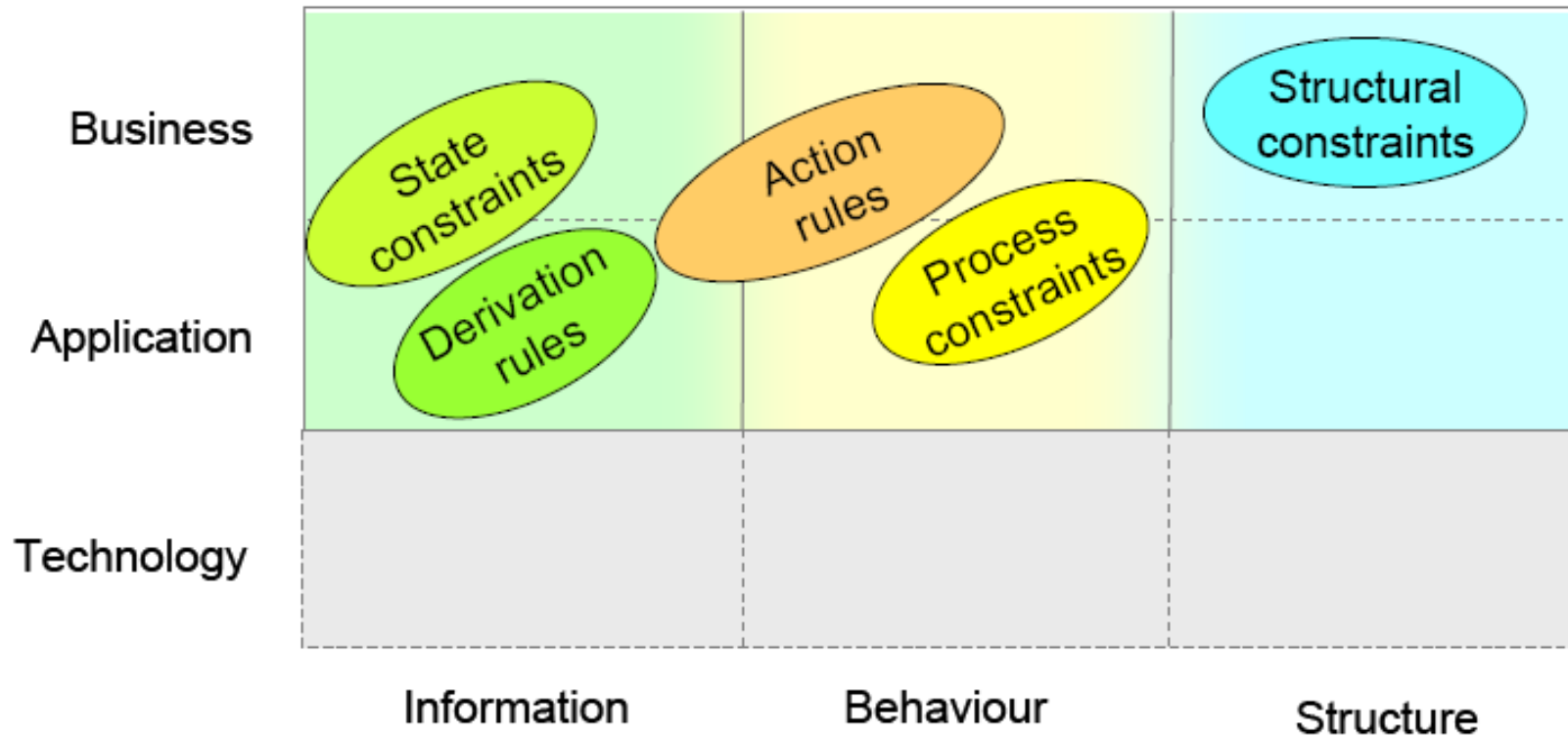
Service getService(int value) {
    for (int i=0;i<getAllServices();i++) {
        Service s = getService(i);
        if (getAvailability(s)==1) && (value==1) return s;
        else if (getAvailability(s)<0,98) && (value < 0,98) return s;
        else return null;
    }
}

...

int getDiscount(Service s) {
    if (getAvailability(s)==1) return 5;
    else if (getAvailability(s)<0,98) return -5;
    else return 0;
}

...
```

Arten von Web Regeln



Web Regeln

1. Regeln

- **Derivation rules** (Deduktionsregeln): Ableitung neues Wissens welches in z.B. Entscheidungsprozessen verwendet wird
- **Reaction rules**: geben an, wann bestimmte Aktionen stattfinden sollen:
 - *Condition-Action rules* (**production rules**)
 - *Event-Condition-Action* (**ECA**) **rules** + Varianten (z.B. ECAP).
 - *Messaging reaction rules* (Ereignisnachrichten und Aktionsnachrichten)

2. (Integrity) Constraints

- *Structural constraints* (deontic assignments).
- *State constraints*.
- *Process / flow constraints*.

Web Rule Languages

Computational
Independent

SBVR

Platform
Independent

PRR

RuleML

RIF

OCL

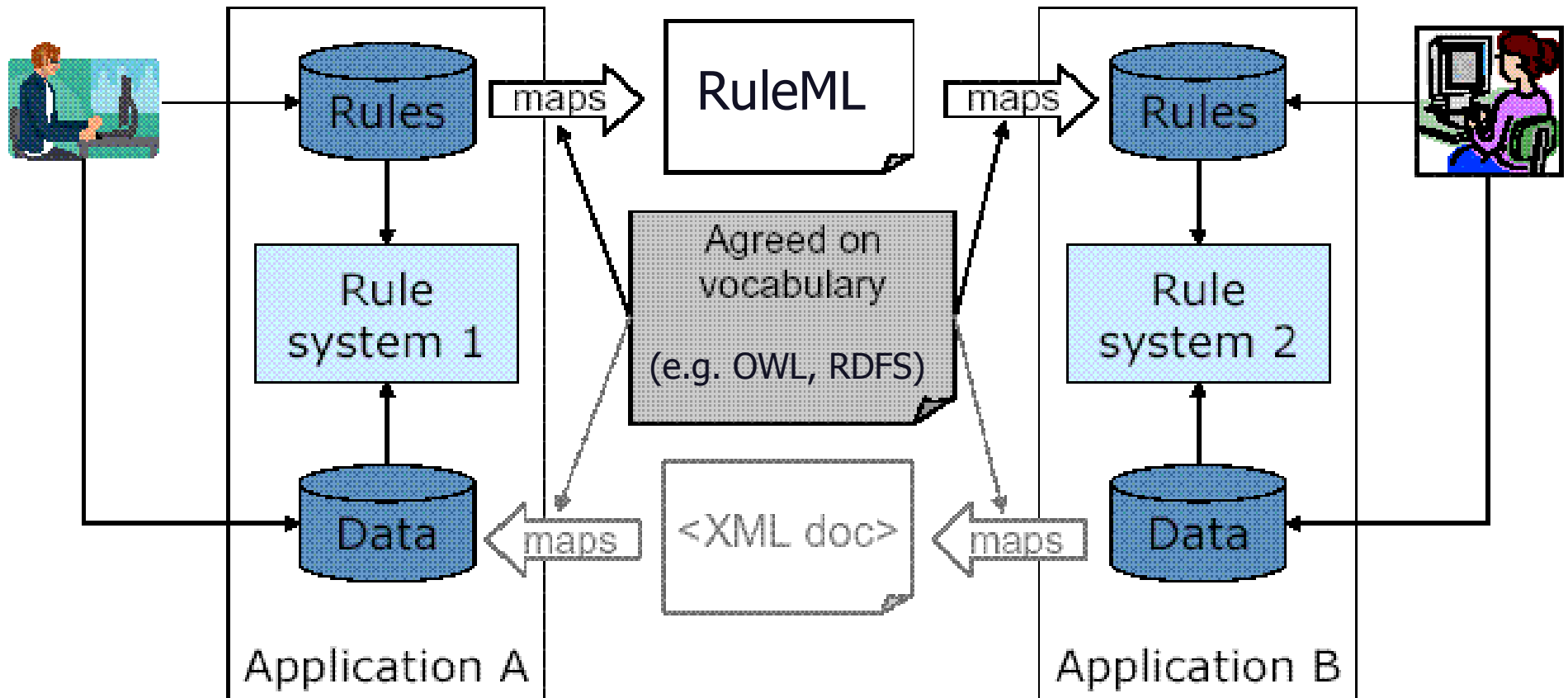
Platform
Specific

ILog **Blaze**
IRL **SRL**

Prova

XCML

Regelaustausch und Regelpublikation



RuleML

RuleML

- Rule Markup and Modeling Initiative (RuleML)
 - <http://www.ruleml.org>
- Förderung moderner und zukünftiger Generationen von Web-basierter Regeltechnologien
- Offener de facto **Sprachstandard für Web Regeln**
 - Basis für weitere Regelsprachen z.B. OMG PRR, W3C RIF, W3C SWRL, REVERSE R2ML,

RuleML ermöglicht ...

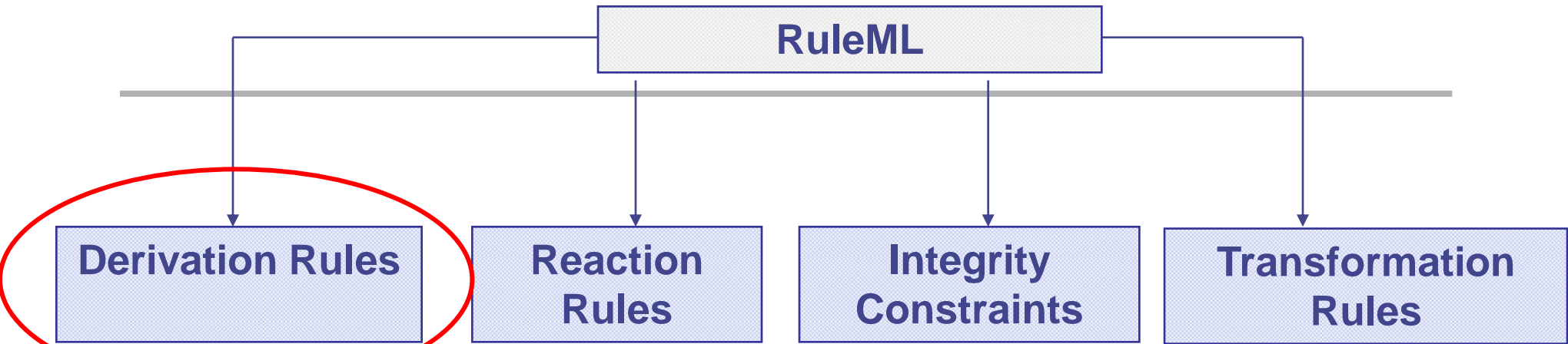
Rule

modelling
markup
translation
interchange
execution
publication
archiving

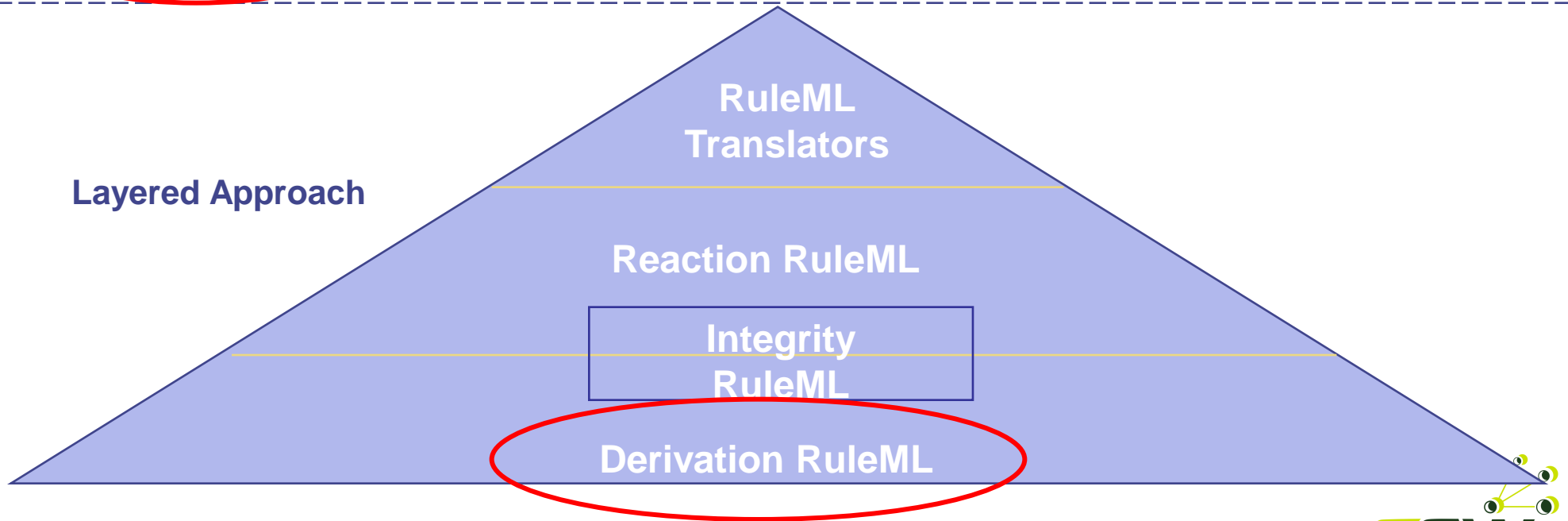
in

UML
RDF
XML
ASCII

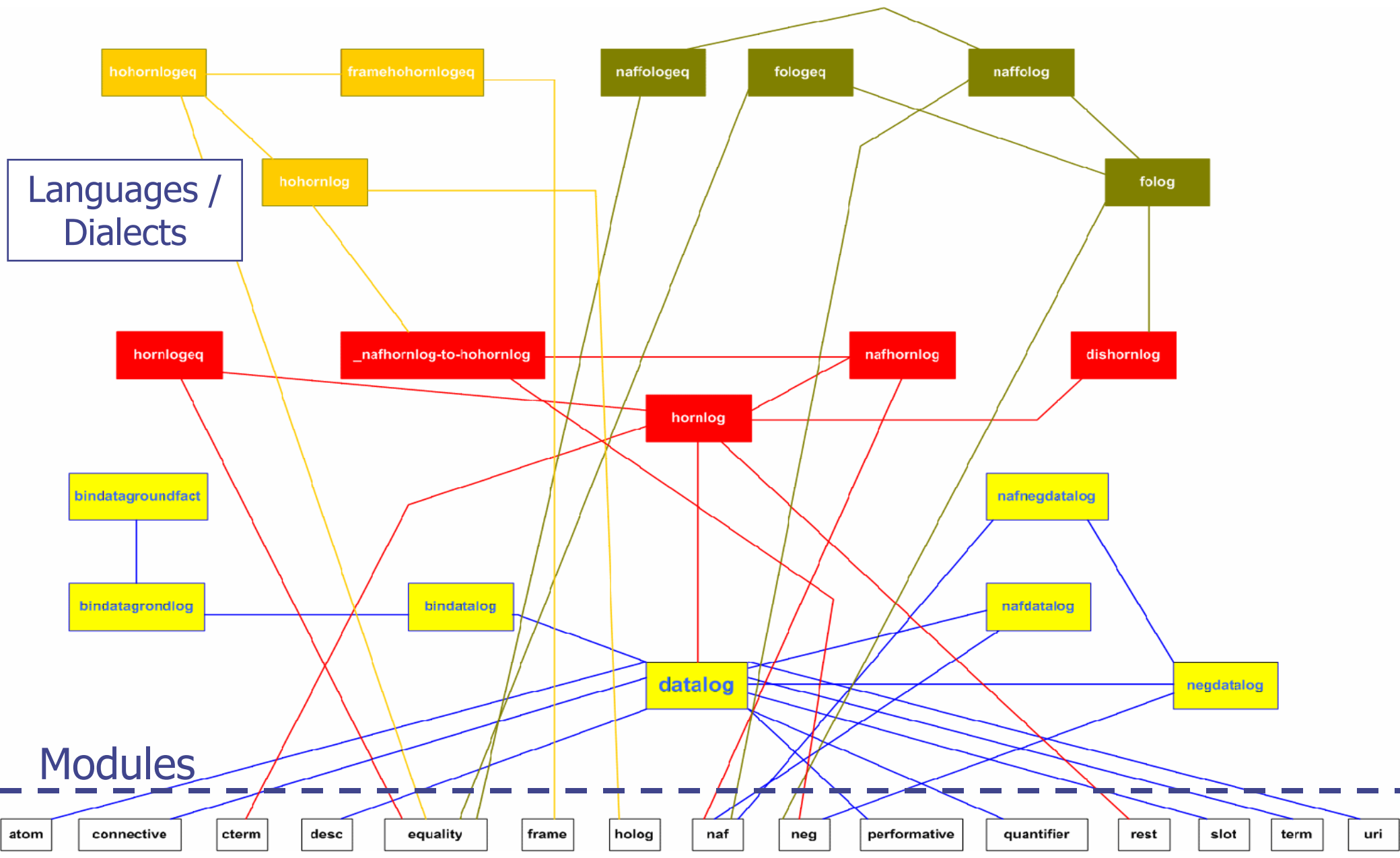
RuleML Sprachfamilie



Layered Approach

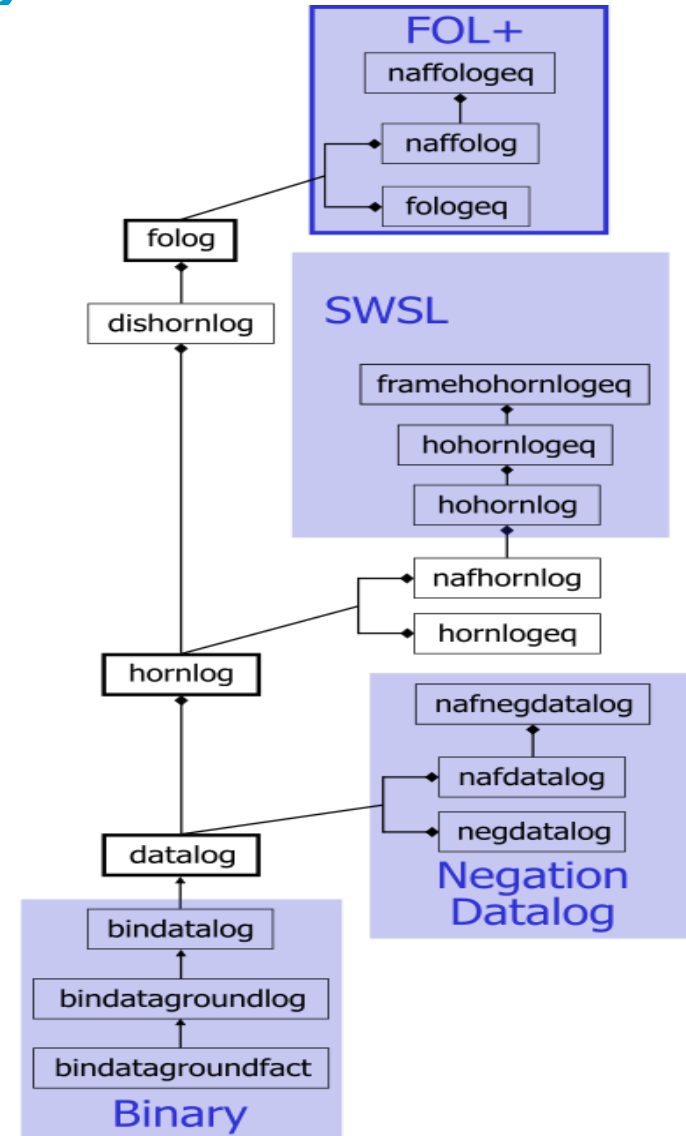


RuleML Sprachfamilie – Derivation RuleML



Schema Modularisierung

- RuleML ist durch modulare XSDs spezifiziert
- XML Schema + EBNF Syntax
- Voll RDF kompatibel (~ RDF triple syntax);
- XML Schema Modularisierung:
Feinschichtiges und einheitliches Design



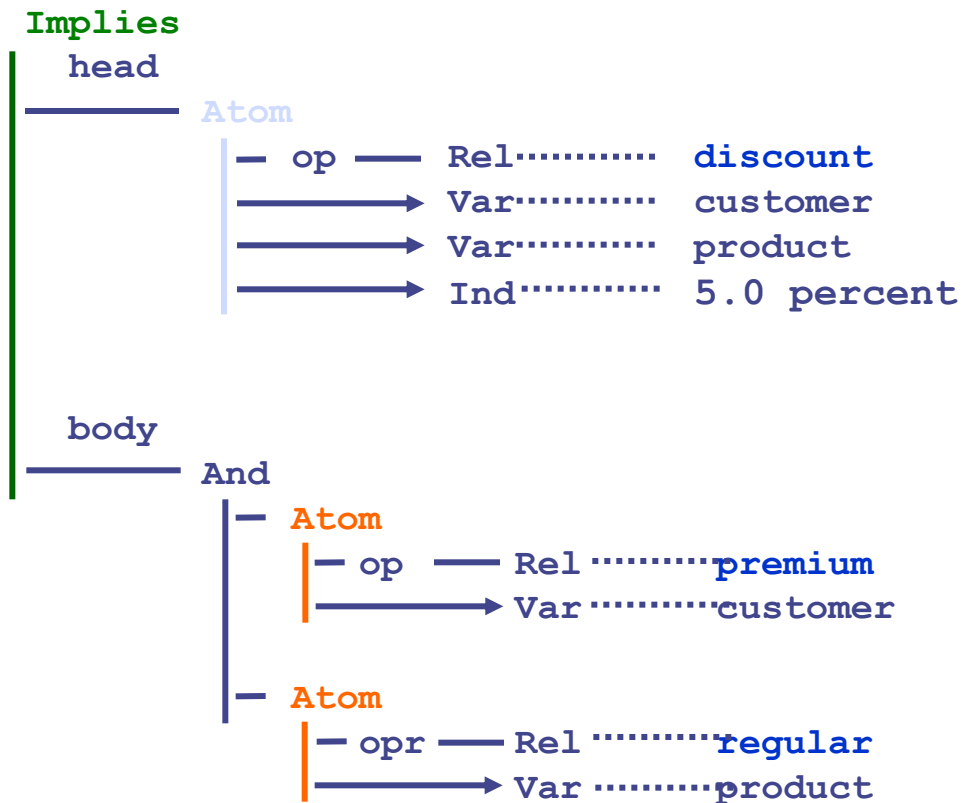
RuleML 0.91 – Striped Syntax

"The **discount** for a *customer* buying a *product* is **5.0 percent** if the *customer* is **premium** and the *product* is **regular**."

```

<Implies>
  <head>
    <Atom>
      <op><Rel>discount</Rel></op>
      <Var>customer</Var>
      <Var>product</Var>
      <Ind>5.0 percent</Ind>
    </Atom>
  </head>
  <body>
    <And>
      <Atom>
        <op><Rel>premium</Rel></op>
        <Var>customer</Var>
      </Atom>
      <Atom>
        <op><Rel>regular</Rel></op>
        <Var>product</Var>
      </Atom>
    </And>
  </body>
</Implies>
  
```

Type Tag
Role Tag



Striped Syntax vs. Stripe-skipped Syntax

```
<Implies>
  <body>
    <Atom>
      <Rel>spending</Rel>
      <Var>customer</Var>
      <Ind>min 5000 euro</Ind>
      <Ind>previous year</Ind>
    </Atom>
  </body>
  <head>
    <Atom>
      <Rel>premium</Rel>
      <Var>customer</Var>
    </Atom>
  </head>
</Implies>
```

```
<Implies>
  <Atom>
    <Rel>spending</Rel>
    <Var>customer</Var>
    <Ind>min 5000 euro</Ind>
    <Ind>previous year</Ind>
  </Atom>
  <Atom>
    <Rel>premium</Rel>
    <Var>customer</Var>
  </Atom>
</Implies>
```

RuleML 0.91 - Constants

- Logische Konstanten-Terme werden durch `<Ind>` Tags in RuleML

Prova

RuleML

abc

`<Ind>abc</Ind>`

“Adrian Paschke”

`<Ind>Adrian Paschke</Ind>`

42

`<Ind>42</Ind>`

RuleML 0.91 - Variables

- Logische Variablen-Terme werden durch `<Var>` Tags in RuleML ausgedrückt

Prova

RuleML

X

`<Var>X</Var>`

—

`<Var/>`

RuleML 0.91 - Functions

- Funktionsausdrücke / Komplexe Term werden durch das `<Expr>` tag ausgedrückt
- Der Funktionsname ist repräsentiert durch das eingebettete `<Fun>` Tag
- Argumente können sein `<Ind>`, `<Var>`, `<Expr>`, `<Plex>`

Prova

$f(X)$

RuleML

```
<Expr>  
  <Fun>f</Fun>  
  <Var>X</Var>  
</Expr>
```

RuleML 0.91 - Lists

- Listen werden durch das `<Plex>` Tag repräsentiert

Prova

[p,1,2]

p(1,2)

RuleML

`<Plex>`

`<Rel>p</Rel>`

`<Ind>1</Ind>`

`<Ind>2</Ind>`

`</Plex>`

RuleML 0.91 - Lists

- Das RuleML `<repo>` (rest, positional) Tag für den Prolog/Prova “|” Operator

Prova

[Head | Rest]

RuleML

`<Plex>`

`<Var>Head</Var>`

`<repo>`

`<Var>Rest</Var>`

`</repo>`

`</Plex>`

RuleML 0.91 - Equality

- Ein Gleichheitsformel bestehend aus zwei Ausdrücken wird durch das `<Equal>` Tag ausgedrückt

Prova

$X=1$

RuleML

```
<Equal>  
  <Var>X</Var>  
  <Ind>1</Ind>  
</Expr>
```

RuleML-0.91 Types

- Typen können Termen über das type Attribute zugeordnet werden
- Typen sind zulässig für <Ind>, <Var> und <Expr> terms

<Var type="Vehicle">Car</Var>

<Ind type="Sedan ">2000 Toyota Corolla</Ind>

RuleML 0.91 – Atomic Formula

- Atomare Formeln durch das <Atom> Tag
- Relationsname ist durch das eingebettet <Rel> Tag angegeben
- Argumenten sind in <Atom> eingebettet – sie können sein <Ind>, <Var>, <Expr>, und <Plex>

Prova

spending(“Peter Miller”, “min 500 euro”, “previous year”).

RuleML

<Atom>

<Rel>spending</Rel>

<Ind>Peter Miller</Ind>

<Ind>min 500 euro</Ind>

<Ind>previous year</Ind>

</Atom>

RuleML 0.91 –Derivation Rules

- (Derivation) Rules sind durch das <Implies> Tag dargestellt
- Erste Kindelement ist der Körper (body) der Regel – kann entweder ein einfaches <Atom> oder eine Konjunktion von <Atom>s in einem <And> Tag sein
- Zweites Kindelement ist der Kopf (head) der Regel, welcher eine atomare Formel (Atom) sein muss

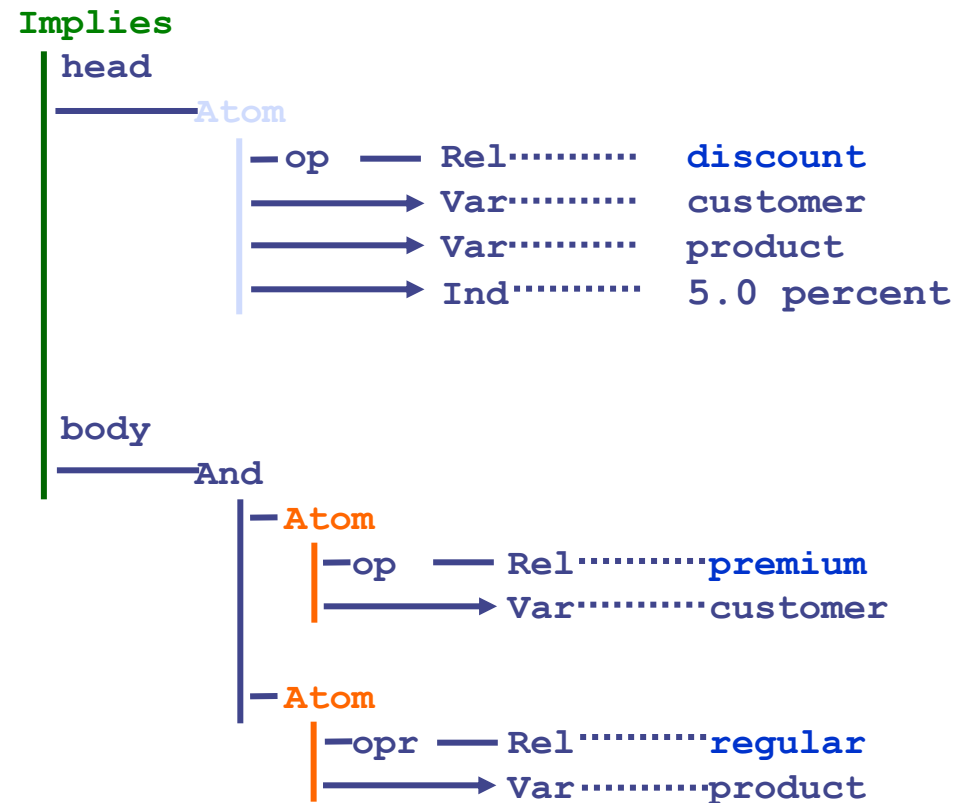
RuleML 0.91 – Positional Representation

"The **discount** for a *customer* buying a *product* is **5.0 percent** if the *customer* is **premium** and the *product* is **regular**."

```

<Implies>
  <head>
    <Atom>
      <op><Rel>discount</Rel></op>
      <Var>customer</Var>
      <Var>product</Var>
      <Ind>5.0 percent</Ind>
    </Atom>
  </head>
  <body>
    <And>
      <Atom>
        <op><Rel>premium</Rel></op>
        <Var>customer</Var>
      </Atom>
      <Atom>
        <op><Rel>regular</Rel></op>
        <Var>product</Var>
      </Atom>
    </And>
  </body>
</Implies>
  
```

Type Tag
Role Tag



RuleML 0.91 – Knowledge Bases

- Eine RuleML Wissensbasis ist eine Konjunktion von Sätzen (Regeln und Fakten), die als wahr hinzugefügt (asserted) werden

<Assert>

<And innerclose="universal">

<!-- Clauses (Facts and Rules) here -->

</And>

</Assert>

RuleML 0.91 – Queries

- Anfragen werden repräsentiert durch das <Query> Tag
- Enthält ein Kind – dies ist ein <Atom> oder eine Konjunktion von <Atom>s in einem <And>
- Beispiel: solve(**premium(Who)**)

```
<Query>  
  <Atom>  
    <Rel>premium</Rel>  
    <Var>who</Var>  
  </Atom>  
</Query>
```

Slotted Un-positional Object Oriented Representation

- Positionsunabhängige Benutzer-Definierte role -> filler Paare:

<Implies>

<Atom>

<Rel>spending</Rel>

<slot><Ind>spender</Ind><Var>customer</Var></slot>

<slot><Ind>amount</Ind><Ind>min 5000 euro</Ind></slot>

<slot><Ind>period</Ind><Ind>previous year</Ind></slot>

</Atom>

<Atom>

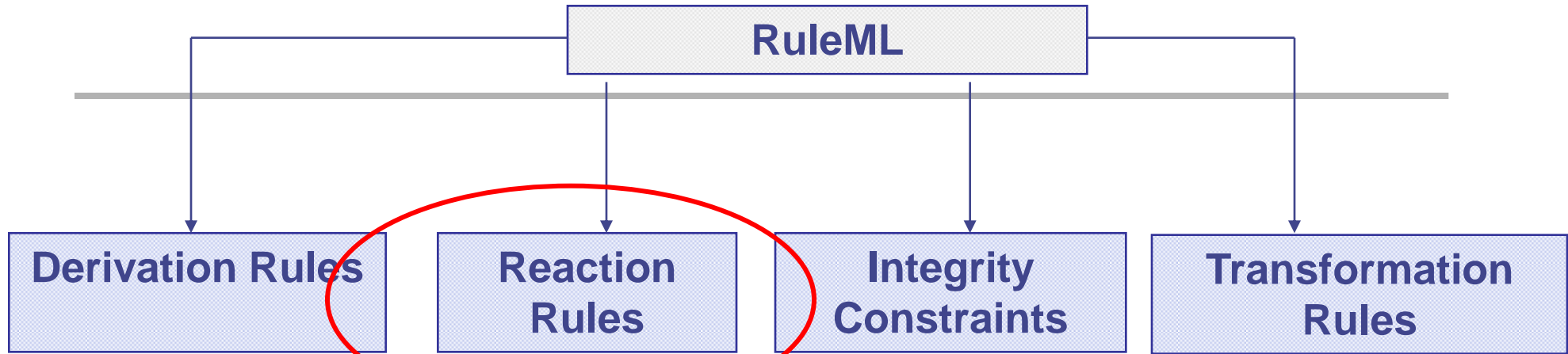
<Rel>premium</Rel>

<slot><Ind>client</Ind><Var>customer</Var></slot>

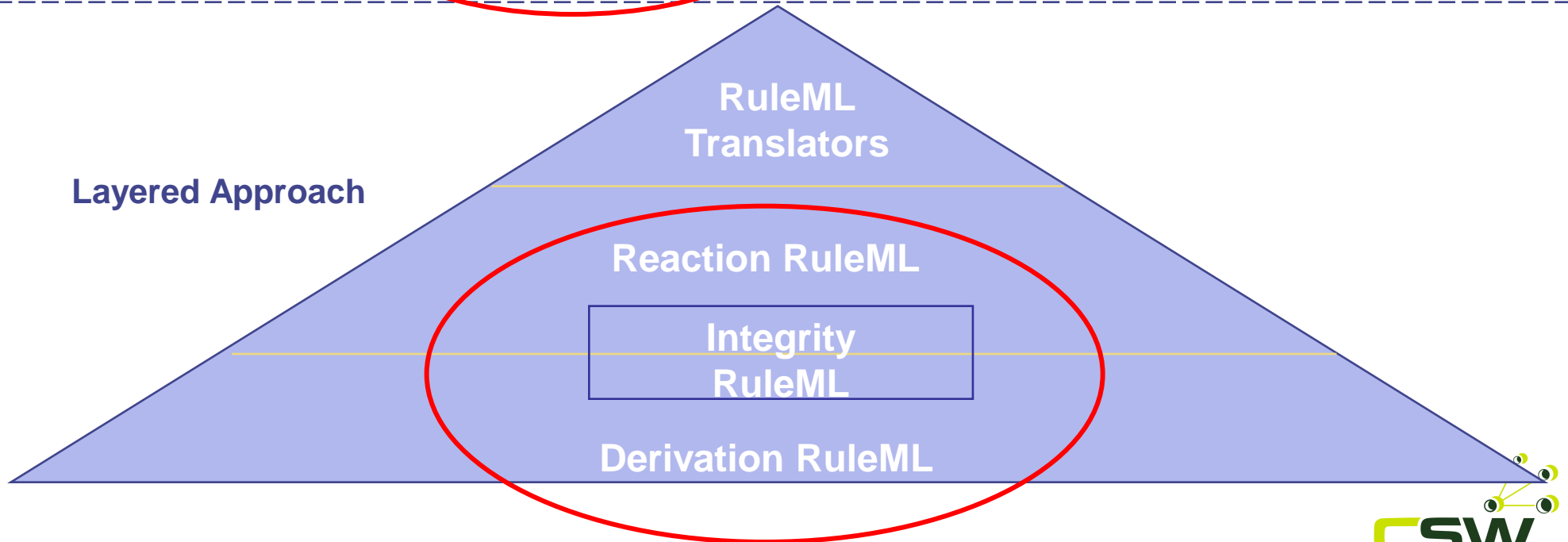
</Atom>

</Implies>

RuleML Sprachfamilie



Layered Approach



Reaction RuleML

- Reaction RuleML (<http://ibis.in.tum.de/research/ReactionRuleML/>)
- **Quasi-Standard for Reactive Web Rules**
 - Production rules, ECA rules and variants such as Trigger (EA), intelligent rule-based CEP, KR Event/Action Logics, Process Algebras, ...
- Anwendungen z.B. in.:
 - Event Processing Networks (EPN)
 - Complex Event Processing (CEP) / Event Processing Language (EPL)
 - Event Driven Architectures (EDAs)
 - Reactive, rule-based Service-Oriented Architectures (SOAs)
 - Active Semantic Web Applications
 - Real-Time Enterprise (RTE)
 - Business Activity Management (BAM)
 - Business Performance Management (BPM)
 - Service Level Management (SLM) with active monitoring and enforcing of Service Level Agreements (SLAs) or e-Contracts
 - Supply Chain Event Management
 - Policies
 - Web-based Workflow Systems
 - ...

Anwendungsbereich

Reaction RuleML

Active Databases

Production Rule
Systems

Rule-Based Event Notification
Systems / Distributed Complex
Event Processing

KR Event / Action /
Transition / Process
Logic Systems

- * Transient Events
- * ECA Paradigm
- * Global Active Rules
- * Trigger (EA Rules)
- * Complex Event Algebra

- * Implicit Sequence of
Knowledge Updates
- * CA Rules

- * Event / Action Messages
 - Inbound / Outbound
 - Enterprise Service Bus
- * (Agent) Conversation
 - Protocols
 - Performatives (e.g. FIPA ACL)

- * Event / Action Axioms
- * Reasoning on Effects /
Transitions
 - fluents / states / processes
 - akin to e.g. state machines,
petri-nets or pi-calculus

Generelle Konzepte

- Generelles Regelformat, welches je nach Bedarf spezialisiert werden kann
- Drei generelle Ausführungsarten (**styles**):
 - **Active**: aktive Abfrage / Erkennung auftretender Ereignisse (events) z.B. regelmäßige Abfrage eines Dienstes oder Anfrage an Ereignisdatenbank
 - **Messaging**: Warten auf eingehende Ereignisnachrichten und Aussendung von Aktionsnachrichten
 - **Reasoning**: logische Event/Action Logics, Transaktionslogiken, Prozesskalküle, z.B. Event Calculus, Temporal Action Logics
- Vorkommen (**appearance**):
 - **Global**: globale Reaktionsregeln
 - **Local**: lokal definierte (inline) Reaktionsregeln

Syntax für Reaktionsregeln (Reaction RuleML 0.2)

```
<Rule style="active" evaluation="strong">
  <label> <!-- metadata --> </label>
  <scope> <!-- scope --> </scope>
  <qualification> <!-- qualifications --> </qualification>
  <oid> <!-- object identifier --> </oid>
  <on> <!-- event --> </on>
  <if> <!-- condition --> </if>
  <then> <!-- conclusion --> </then>
  <do> <!-- action --> </do>
  <after> <!-- postcondition --> </after>
  <else> <!-- else conclusion --> </else>
  <elseDo> <!-- else/alternative action --> </elseDo>
  <elseAfter> <!-- else postcondition --> </elseAfter>
</Rule>
```

Reaction RuleML – Regeltypen

- **Derivation Rule:**

```
<Rule style="reasoning">  
  <if>...</if>  
  <then>...</then>  
</Rule>
```
- **Production Rule:**

```
<Rule style="active">  
  <if>...</if>  
  <do>...</do>  
</Rule>
```
- **ECA Rule:**

```
<Rule style="active">  
  <on>...</on>  
  <if>...</if>  
  <do>...</do>  
</Rule>
```

Beispiel Produktionsregel

```
<Rule style="active"> <!-- production rule -->
  <bf>
    <And>
      <Atom><Rel>available</Rel><Var>Service</Var></Atom>
      <Atom><Rel>request</Rel><Var>Task</Var></Atom>
    </And>
  </bf>
  <do>
    <Assert>
      <Atom>
        <Rel>loaded</Rel>
        <Var>Service</Var><Var>Task</Var>
      </Atom>
    </Assert>
  </do>
</Rule>
```

Nachrichten in Reaction RuleML

```
<Message mode="outbound" directive="ACL:inform">
  <oid> <!-- conversation ID--> </oid>
  <protocol> <!-- transport protocol --> </protocol>
  <sender> <!-- sender agent/service --> </sender>
  <content> <!-- message payload --> </content>
</Message>
```

- **@mode** = inbound|outbound – attribute defining the type of a message
- **@directive** – attribute defining the pragmatic context of the message, e.g. a FIPA ACL performative
- **< oid >** – the conversation id used to distinguish multiple conversations and conversation states
- **< protocol >** – a transport protocol such as HTTP, JMS, SOAP, Jade, Enterprise Service Bus (ESB) ...
- **< sender >< receiver >** – the sender/receiver agent/service of the message
- **< content >** – message payload transporting a RuleML / Reaction RuleML query, answer or rule base

Beispiel: Request / Query

```
...  
<Message mode="outbound" directive="ACL:query-ref">  
  <oid> <Ind>RuleML-2008</Ind> </oid>  
  <protocol> <Ind>esb</Ind> </protocol>  
  <sender> <Ind>User</Ind> </sender>  
  <content>  
    <Atom>  
      <Rel>getContact</Rel>  
      <Ind>Sponsoring</Ind>  
      <Var>Contact</Var>  
    </Atom>  
  </content>  
</Message>
```

FIPA ACL directive



- Ereignisnachricht ist lokal zum Konversationsstatus (oid) und dem pragmatischen Kontext (directive)

Komplexe Event/Action Operatoren

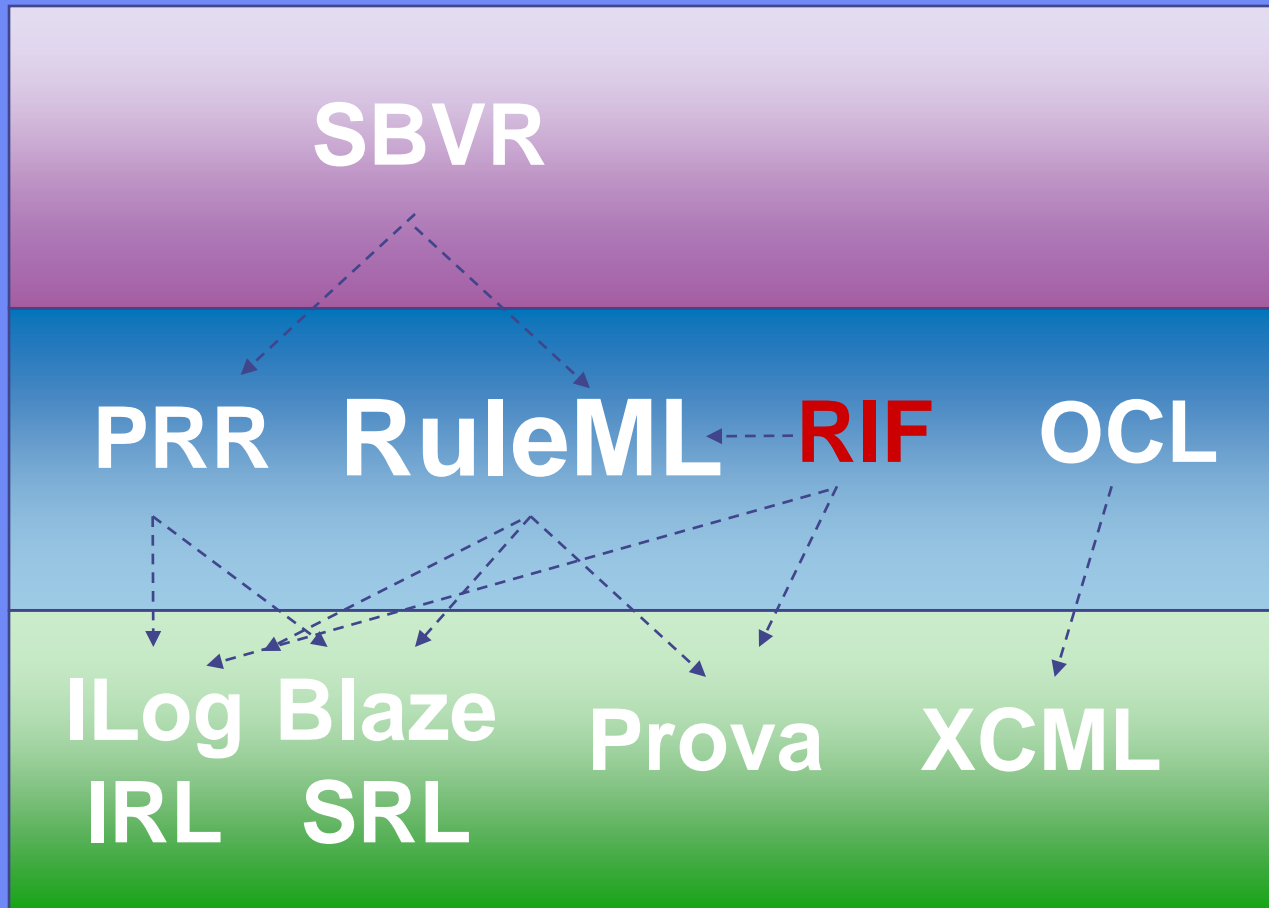
- Action Algebra:
Succession (Ordered Succession of Actions), *Choice* (Non-Deterministic Choice), *Flow* (Parallel Flow), *Loop* (Loops)
- Event Algebra:
Sequence (Ordered), *Disjunction* (Or) , *Xor* (Mutal Exclusive), *Conjunction* (And), *Concurrent* , *Not*, *Any*, *Aperiodic*, *Periodic*

```
<event>
  <Sequence>
    <Concurrent>
      <Ind>a</Ind> <Ind>b</Ind>
    </Concurrent>
    <Ind>c</Ind>
  </Sequence>
</event>
```

Weitere ausgewählte Eigenschaften

- Complex Event and Action Algebra
- Support for different selection and consumption policies
- Support for intervals (Time, Event)
- Support for situations (States, Fluents)
- Support for external event query languages
- Support for external vocabularies
- Support for external action execution / procedural attachments
- ...

W3C RIF



W3C Rule Interchange Format

- W3C RIF Working Group – seit Dezember 2005
http://www.w3.org/2005/rules/wiki/RIF_Working_Group
- **Ziel:** Standardisierung eines Web-basierten Regelaustauschformats
- Aktuelle Arbeitsentwürfe (Working Drafts):
 - **RIF-FLD** (Framework of Logic Dialects)
 - Generelles Framework für logische RIF Sprachdialekte
 - **RIF-SWC** (RDF and OWL Compatibility)
 - Interoperabilität zwischen RIF und Semantic Web Daten- und Ontologiesprachen (RDF, RDFS, OWL).
 - **RIF-DTB** (Data Types and Builtins)
 - RIF Datentypen und Built-in Funktionen und Prädikate
 - **RIF-BLD** (Basic Logic Dialect)
 - Basis-Logik-Dialekt für logische Regeln (definite Horn rules with equality)
 - **RIF-PRD** (Production Rules Dialect)
 - RIF Produktions Regel Dialekt
 - **RIF-UCR** (Use Cases and Requirements)

W3C RIF Basic Logic Dialect (RIF-BLD)

- Definite Horn Rules +
 - Equality, Slots, Frames, Internationalized Resource Identifiers (IRIs), XML Schema Data Types, ...
- Syntaxes
 - Normative XML Syntax
 - sehr ähnlich zu RuleML XML Syntax
 - Nicht-normative Präsentationssyntax

Beispiel RIF-BLD

RIF condition

```
And (Exists ?Buyer (cpt:purchase(?Buyer ?Seller
                                cpt:book(?Author bks:LeRif)
                                curr:USD(49))
    ?Seller=?Author )
```

XML serialization

```
<And>
  <formula>
    <Exists>
      <declare><Var>Buyer</Var></declare>
      <formula>
        <Atom>
          <op><Const type="&rif;iri">&cpt;purchase</Const></op>
          <args ordered="yes">
            <Var>Buyer</Var>
            <Var>Seller</Var>
            <Expr>
              <op><Const type="&rif;iri">&cpt;book</Const></op>
              <args ordered="yes">
                <Var>Author</Var>
                <Const type="&rif;iri">&bks;LeRif</Const>
              </args>
            </Expr>
            <Expr>
              <op><Const type="&rif;iri">&curr;USD</Const></op>
              <args ordered="yes"><Const type="&xsd;integer">49</Const></args>
            </Expr>
          </args>
        </Atom>
      </formula>
    </Exists>
  </formula>
  <formula>
    <Equal>
      <side><Var>Seller</Var></side>
      <side><Var>Author</Var></side>
    </Equal>
  </formula>
</And>
```

RIF Planung bis Mai 2009

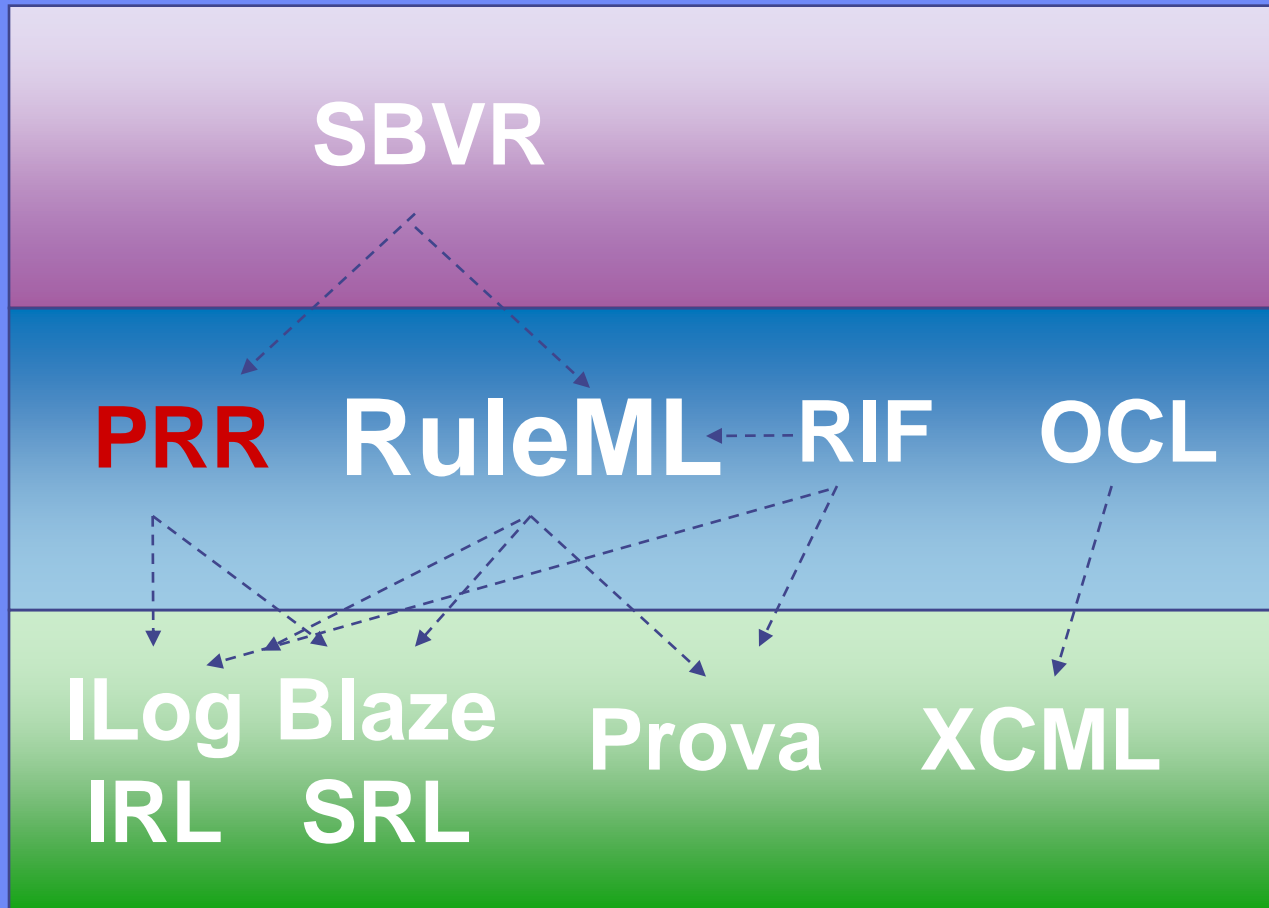
Recomendations:

- RIF Basic Logic Dialect (BLD)
- RIF Framework for Logic Dialects
- RIF Datatypes and Built-Ins
- RIF Production Rule Dialect
- (if possible) Dialect Interoperation and Extensible Core

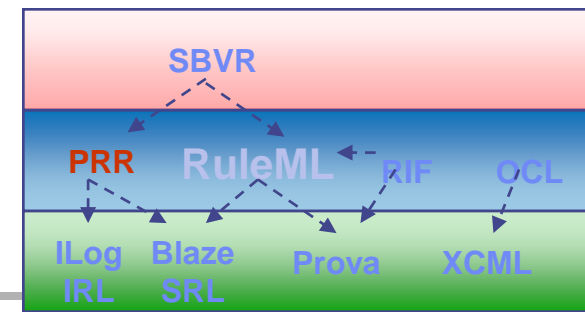
Geplante Working Drafts:

- RIF Logic Programming Dialect
- RIF Full First-Order Logic Dialect
- RIF Extension for XML Data Sources
- RIF Reaction Rules Dialect
 - (includes RIF Event Processing Language Dialect)

OMG PRR



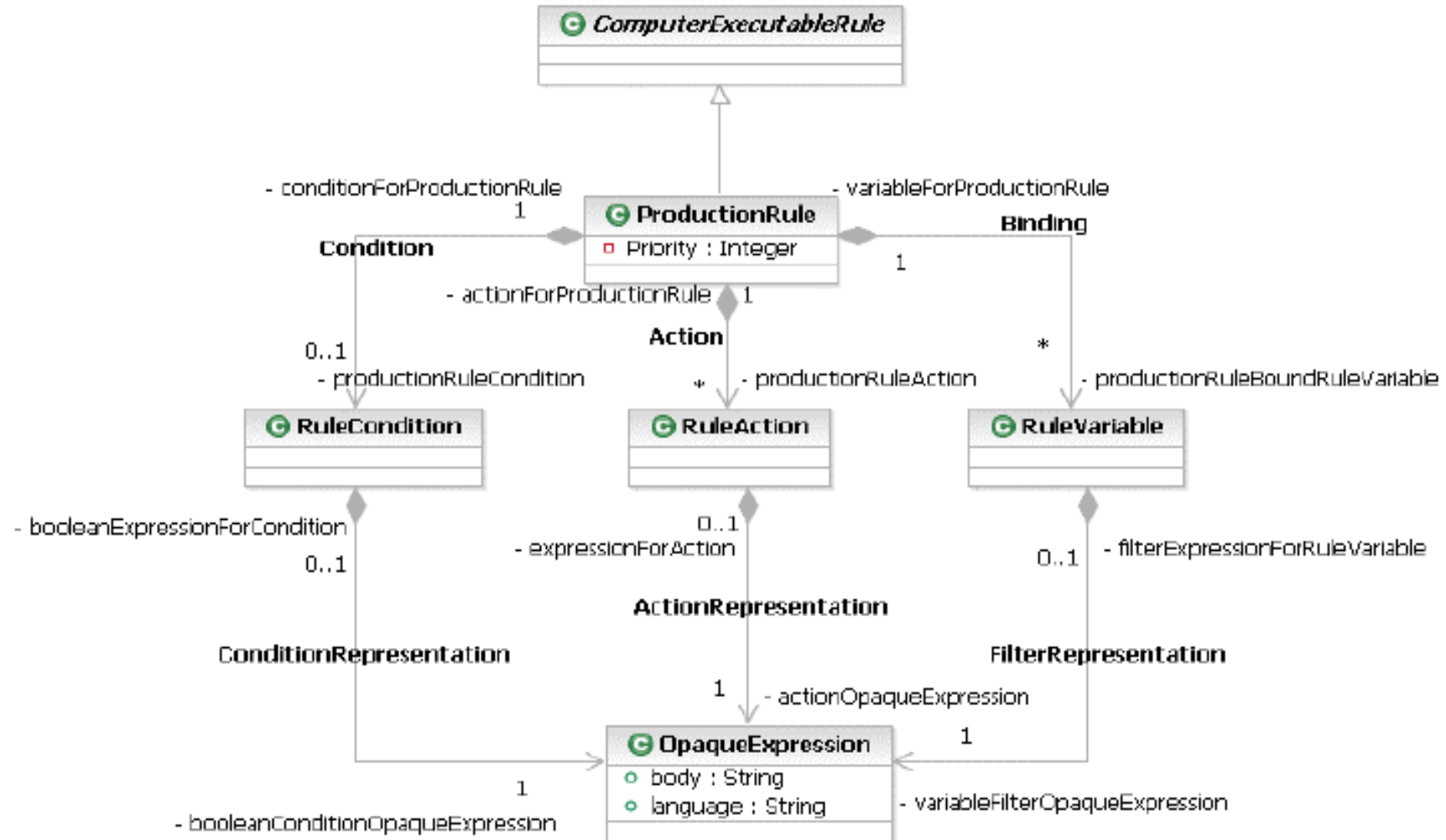
Production Rules Representation



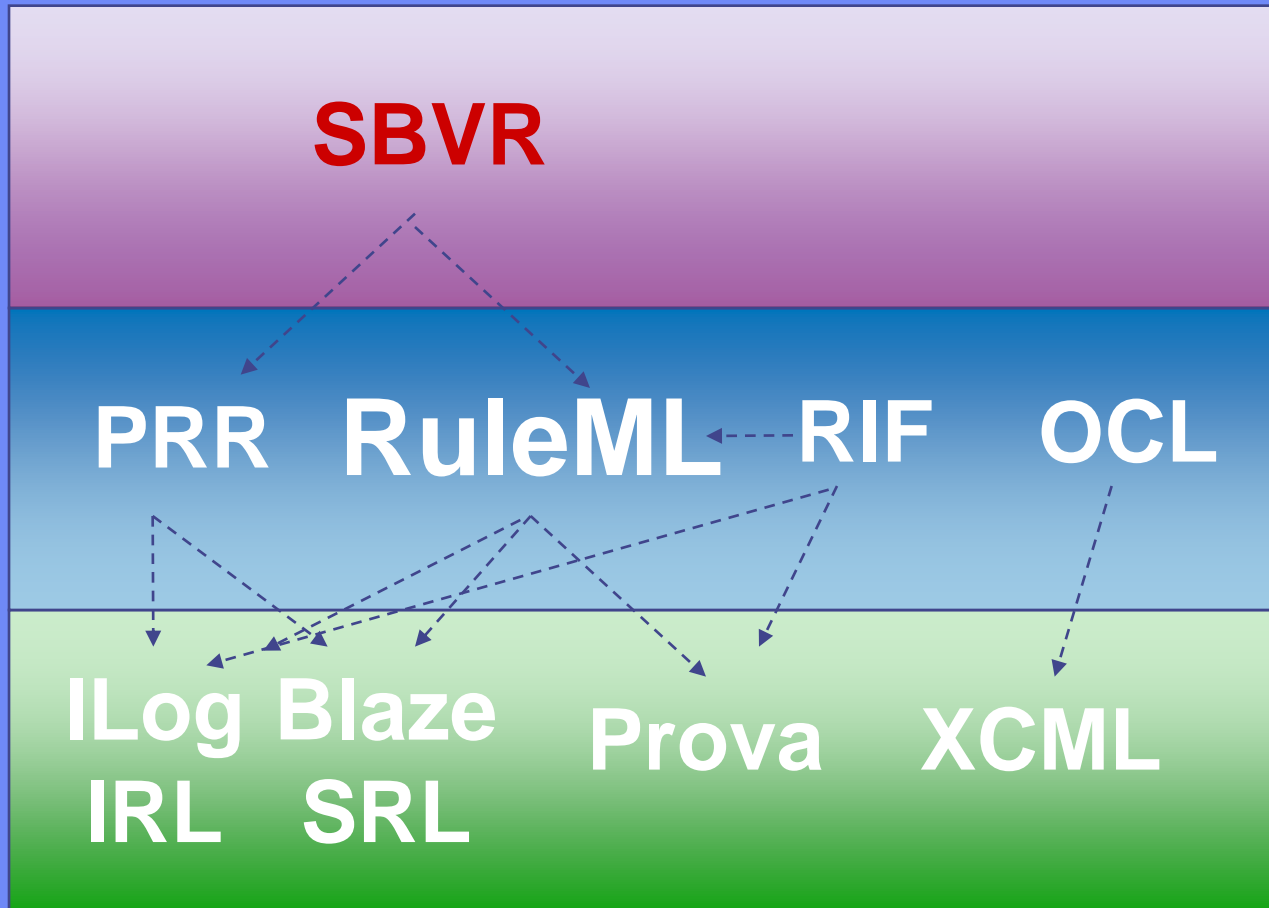
- Formales Model zur Anbieter-neutralen Darstellung von Regelmodellen in UML für Produktionsregeln
- Konsortium von Entwicklern, Anbietern und Benutzern
 - Anbieter (Fair Isaac, ILOG, LibRT, IBM, Pega, Corticon, TIBCO, ...)
 - Akademische Gemeinde (RuleML.org)
 - Verwandte Anbieter (Fujitsu, IBM)
- PRR zur Zeit "Adopted" als Standard in "Finalization",
 - Beta Status, mit finaler Version 1.0 in 2008
- OMG MDA PIM Model
- PRR beta definiert PRR Kern auf Basis von UML MOF
 - Keine explizite Ausdruckssprache
 - (z.B. RuleML Reaction RuleML oder RIF PRD zur Serialisierung)

PRR ProductionRule Klassen

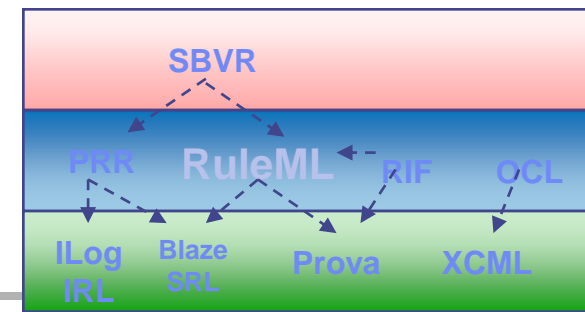
if [condition] then [action-list]



OMG SBVR

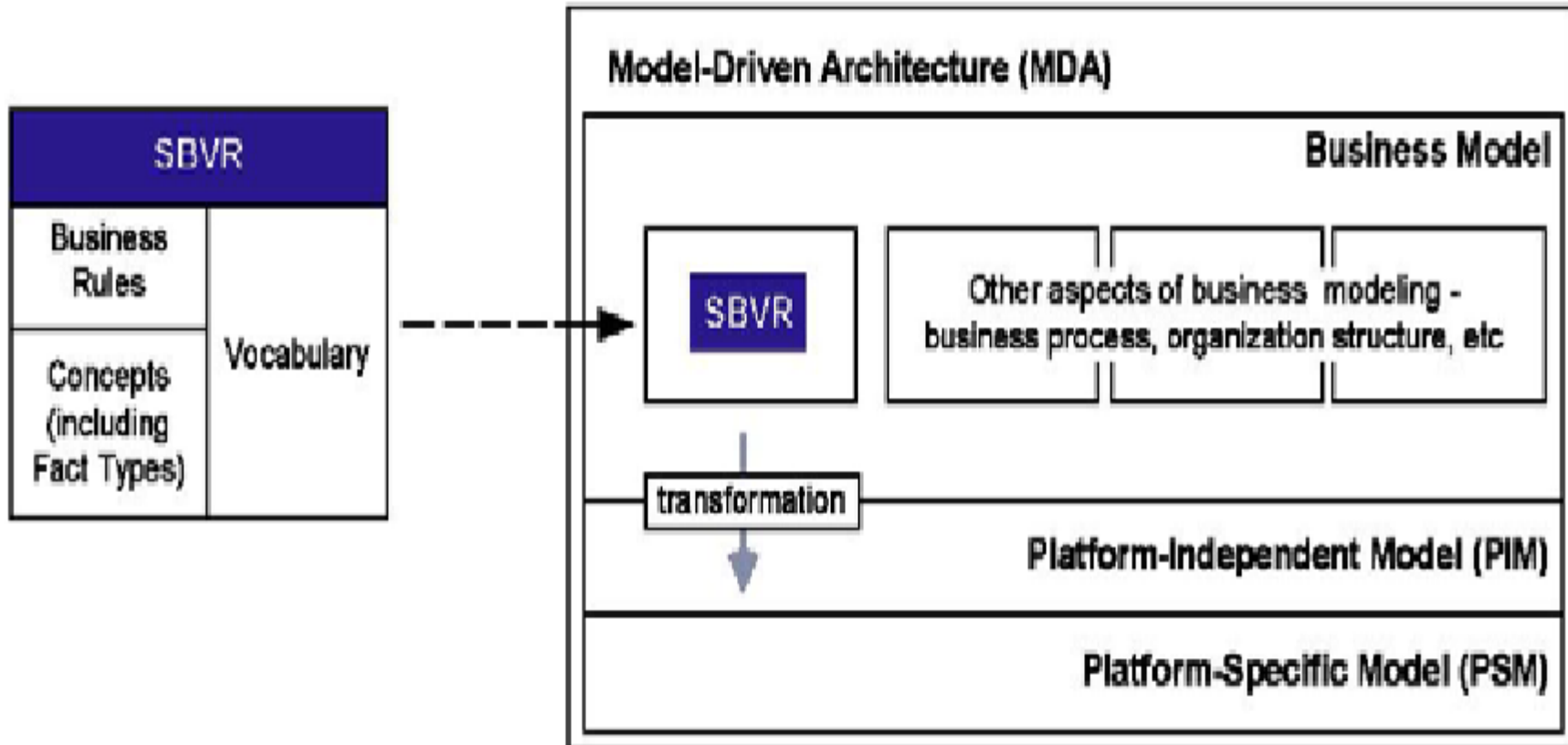


Semantics of Business Vocabulary and Business Rules

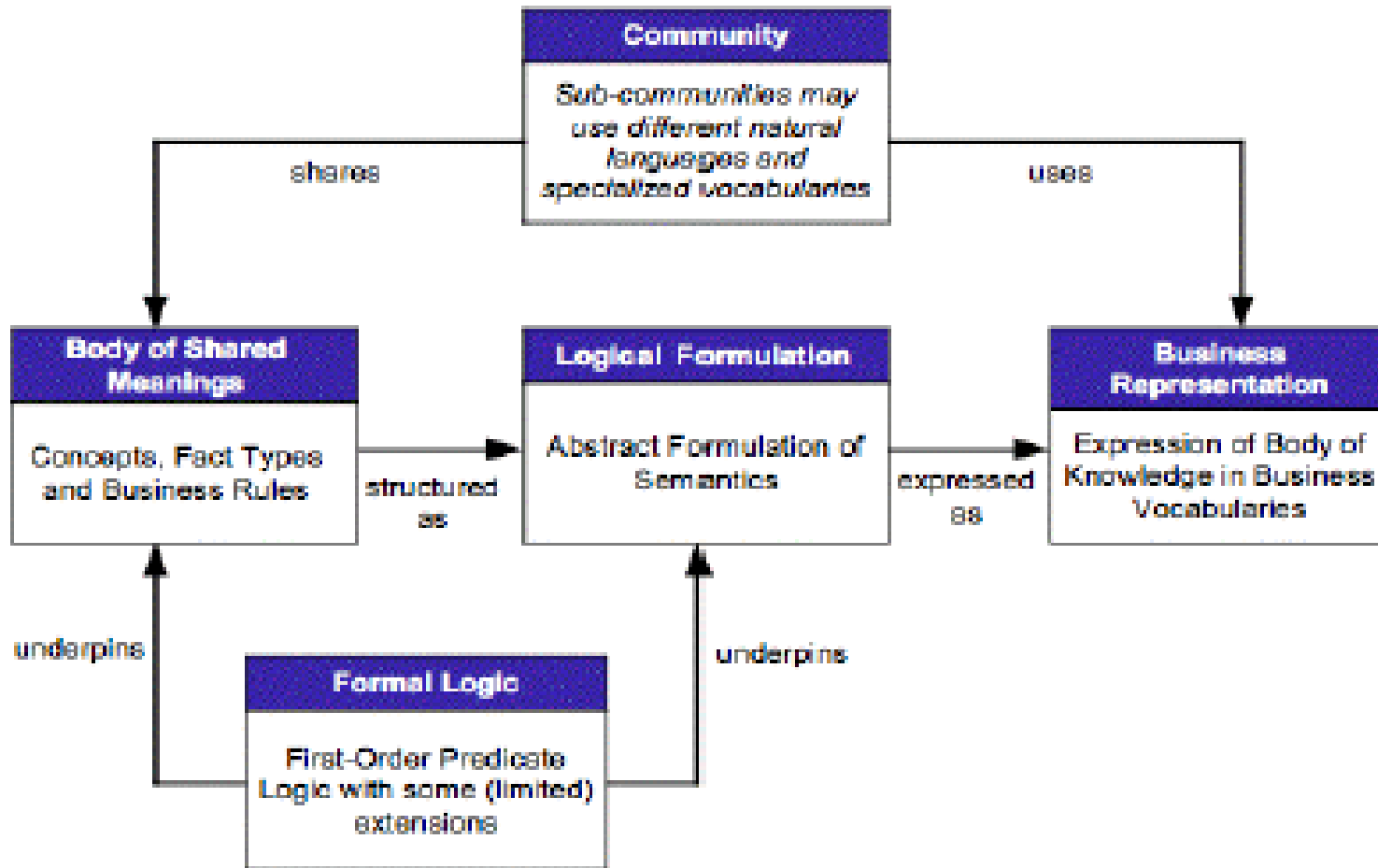


- Unterstützung für Rechenunabhängige Modellierung (computational independent modeling) von Geschäftsregeln (business rules) durch Geschäftsleute — natürliche Sprache, allgemeine Graphiken und Tabellen
- Seit September 2005 „Finalization“ Status im OMG Prozess zur Annahme von Standards
- Final angenommene SBVR 1.0 Spezifikation (dtc/06-03-01)
 - <http://www.omg.org/spec/SBVR/1.0/>

SBVR in MDA



SBVR Überblick

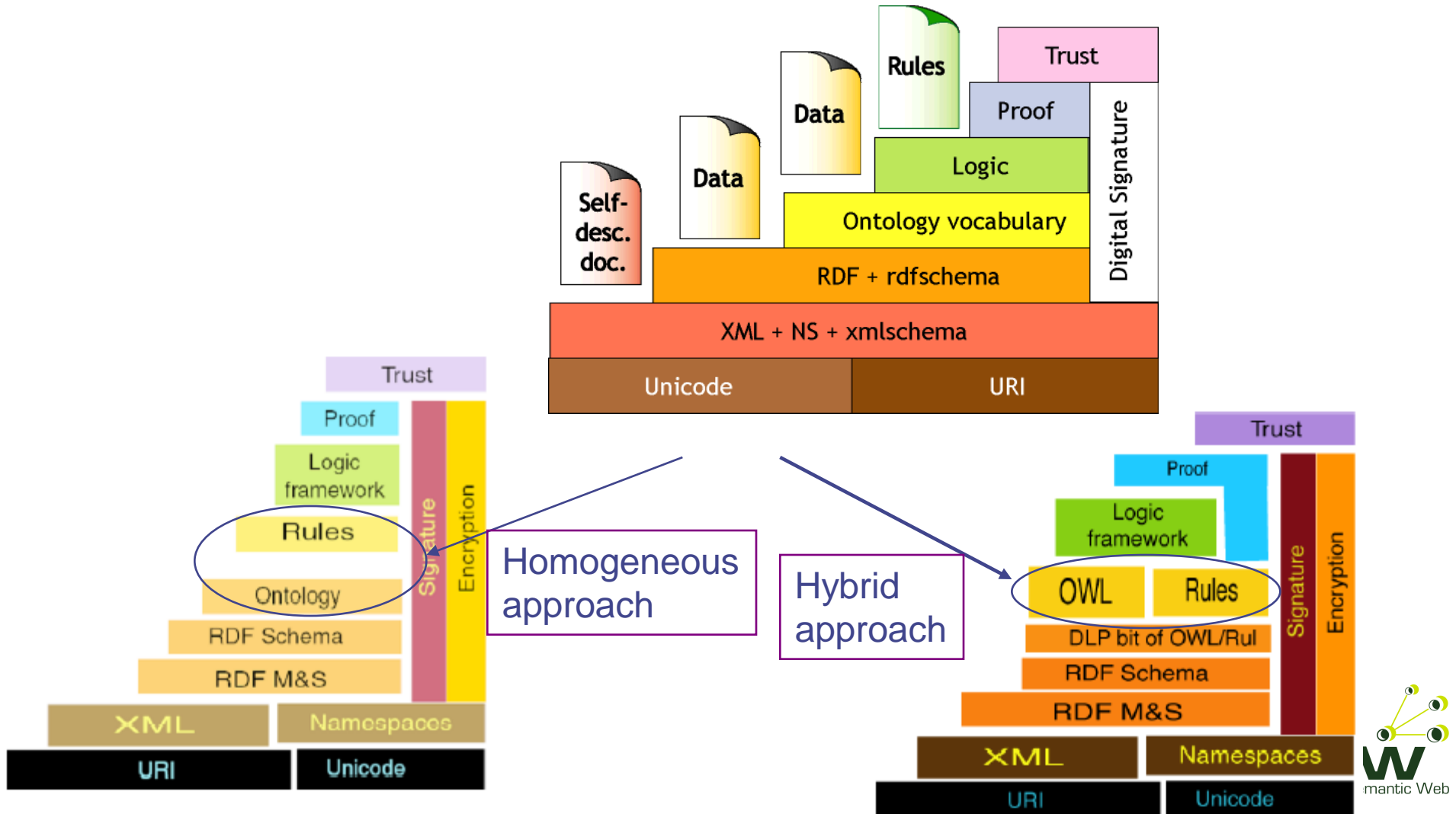


Ontologien

+

Regeln

Web Rules: Semantic Web + Rules



Ontologien + Regeln

- Kombination von Regeln und Ontologien
 - Ziel: Erweiterte Ausdrucksstärke
- Homogene Integration
 - Combination of rules and ontologies (description logics) in one homogenous framework, which uses the same language symbols
 - e.g. DLP, KAON2, SWRL, OWL-R
- Heterogeneous Integration
 - Hybrid approach
 - Heterogeneous integration of DL inference techniques and tools in combination with rule languages and rule engines
 - e.g. CARIN, Life, Prova/ContractLog KR, AI-log, DatalogDL, non-monotonic dl-programs, r-hybrid KBs

SWRL -A Semantic Web Rule Language Combining OWL and RuleML

- Homogene Kombination von RuleML (Datalog Layer) + OWL DL
- SWRL v0.5
- Grundprinzip:
 - Benutze Ontologie-Axiome als Atome in Derivation Rules:

$C(x)$, $P(x,y)$, $\text{sameAs}(x,y)$, $\text{differentFrom}(x,y)$, $\text{builtIn}(r,x,\dots)$

C = OWL description or data range,

P = OWL property

r = built-in relation,

x und y = Variablen oder OWL Individuen oder OWL Datenwerte

- + Homogene Sprache
- Nicht entscheidbar
- Eingeschränkte Ausdrucksmächtigkeit

SWRL 0.5 Example

hasParent(?x1,?x2) ∧ hasBrother(?x2,?x3) ⇒ hasUncle(?x1,?x3)

```
<ruleml:imp>
  <ruleml:_body>
    <swrlx:individualPropertyAtom swrlx:property="hasParent">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x2</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom swrlx:property="hasBrother">
      <ruleml:var>x2</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom swrlx:property="hasUncle">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>
```

OWL2Prova DL-Typed Hybrid Description Logic Programs

- **Hybride Integration**
 - Ontologien in OWL Lite / DL (oder RDFS) in Regeln als Typen für logische Terme
- **Präskriptiver Typisierungansatz: “*term:type*”**
 - Neue Ontologie-Individuen sind in den Regelköpfen erlaubt
 - Freie typisierte Variablen sind als Fakten erlaubt → Interpretation als Anfrage an die *bekannt* Individuen in der A-box
- **Dynamisches Type Checking**
 - Durch Benutzung von externen DL Reasonern für z.B. *subsumption inference*, *instantiate inference*, *equivalence mapping* etc.
 - Caching von vor-inferierten Modellen für schnelleren Zugriff auf externe Ontologie
- **Polymorphic Order-Sorted Unification**
 - Ad-hoc polymorphism: Variablen können ihre Type während der Unifizierung ändern
 - Type-casting im Sinne von “coercion”
- **Unterstützt unterschiedliche Semantic Web Type Systeme:**
 - RDFS, OWL-Lite, OWL-DL , DL ontol.

Beispiel (1)

% Imports

```
import("http://.../dl_typing/businessVocabulary1.owl").  
import("http://.../dl_typing/businessVocabulary2.owl").  
import("http://.../dl_typing/mathVocabulary.owl").  
import("http://.../dl_typing/currencyVocabulary.owl").
```

```
reasoner("dl"). % configure reasoner (OWL-DL=Pellet)
```

% Rule-based Discount Policy

```
discount(X:businessVoc1_Customer, math_Percentage:10) :-  
    gold(X: businessVoc1_Customer).  
discount(X: businessVoc1_Customer, math_Percentage:5) :-  
    silver(X: businessVoc1_Customer).  
discount(X: businessVoc1_Customer, math_Percentage:2) :-  
    bronze(X: businessVoc1_Customer).
```

Beispiel (2)

```
% Note that this rules use a different vocabulary
% if the classes "Client" and "Customer" are equal
% both typed rule sets unify
% Class equivalence between both "types"
% is defined in the second OWL-DL ontology
```

```
gold(X:businessVoc2_Client) :-
  spending(X:businessVoc2_Client, S:currency_Dollar),
  S:currency_Dollar > currency_Dollar:1000.
```

```
silver(X:businessVoc2_Client) :-
  spending(X:businessVoc2_Client, S:currency_Dollar),
  S:currency_Dollar > currency_Dollar:500,
  S:currency_Dollar < currency_Dollar:1000.
```

```
bronze(X:businessVoc2_Client) :-
  spending(X:businessVoc2_Client, S:currency_Dollar),
  S:currency_Dollar > currency_Dollar:100,
  S:currency_Dollar < currency_Dollar:500.
```

Beispiel (3)

% Facts

```
spending (businessVoc1_Customer:Adrian, currency_Dollar:1100) .
```

```
spending (businessVoc2_Client:Aira, currency_Dollar:200) .
```

% Query

```
:-solve (discount (X:businessVoc2_Client,  
Y:math_Percentage) ) .
```

Result:

```
X:businessVoc2_Client = businessVoc1_Customer:Adrian
```

```
Y:math_Percentage = math_Percentage:10
```

```
X:businessVoc2_Client = businessVoc1_Customer:Aira
```

```
Y:math_Percentage = math_Percentage:2
```

Typen in RuleML

- Typen können Terme zugewiesen werden durch Benutzung des “type” Attributes
- Typen können für <Ind>, <Var> und <Cterm> definiert werden
- Nicht jedoch für <Plex>s und <Atom>s
- Beispiel:

```
<Var type="Vehicle" />
```

```
<Ind type="Sedan ">
```

```
2000 Toyota Corolla
```

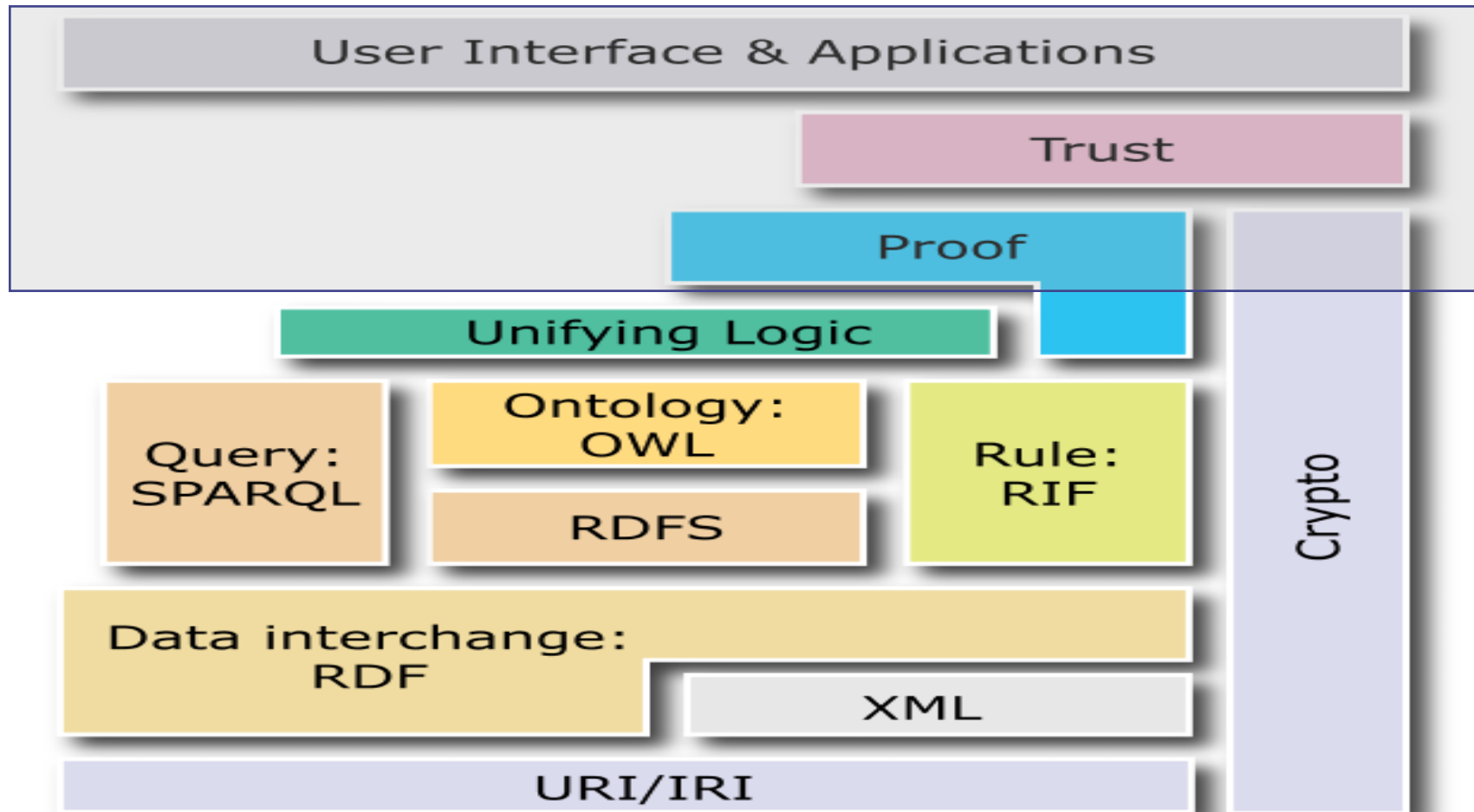
```
</Ind>
```

Vorteile: Heterogene Kombination

- Statisches und Dynamisches Type Checking
- Unterstützt Software Engineering Prinzipien wie Datenabstraktion, Modularisierung etc.
- Reduziert Suchraum auf Anfragen für Regelsets des entsprechenden Types
- Erweitert die Ausdrucksmächtigkeit von Regeln mit Description Logics
- Nutzt hoch optimierte externe DL Reasoner
- Caching-Strategien zwischen inferierter Ontologie und Regelsystem ist möglich
- Erlaubt ad-hoc Polimorphismus mit Überladung und Coercion
- Integration von Domänen-spezifischen Semantic Web Ontologien (Vokabularen) in Domänen-unabhängige Regelspezifikationen
 - Unterstützt Regelaustausch
 - Unterstützt Kollaboration
 - Unterstützt verteiltes Management

The Future of the Semantic Web

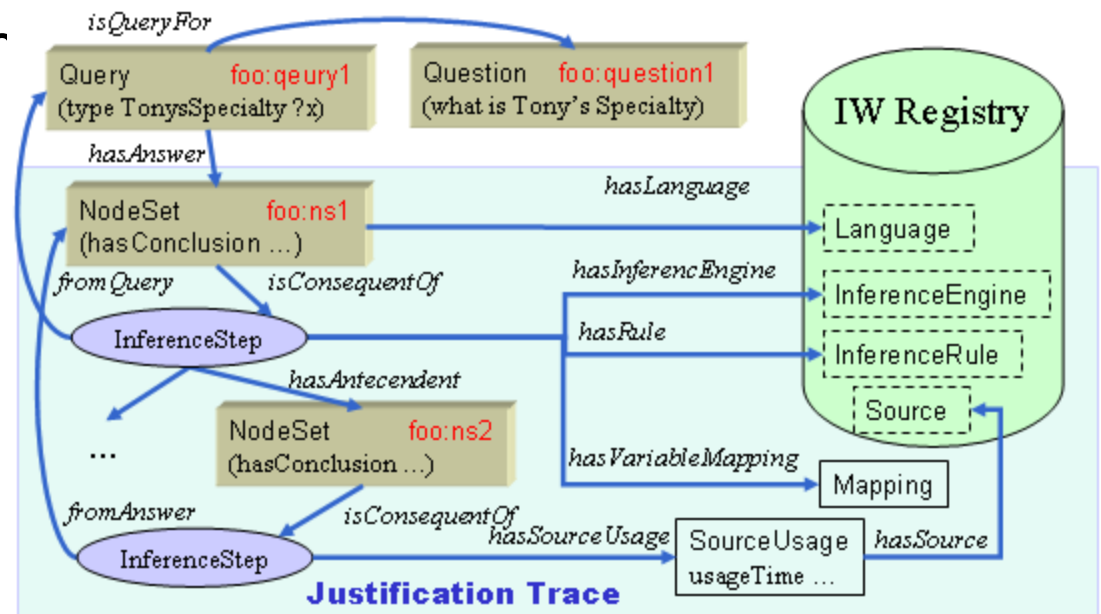
Aktueller Semantic Web Stack



W3C Semantic Web Stack as of 2008

Proof Markup Language

- Proof Markup Language (PML) hilft Benutzern ihre Beweise in RDF zu kodieren und gemeinsam zu nutzen
- Werkzeuge zum Generieren, Suchen und Visualisieren der Beweisinhalte
- PML Ontologie hilft die Rechtfertigungsspur zu kodieren (eine solche Spur kann Benutzern helfen mehrere unterschiedliche Beweisalternativen zu rekonstruieren, welche die mögliche Antwort auf eine Frage sind)

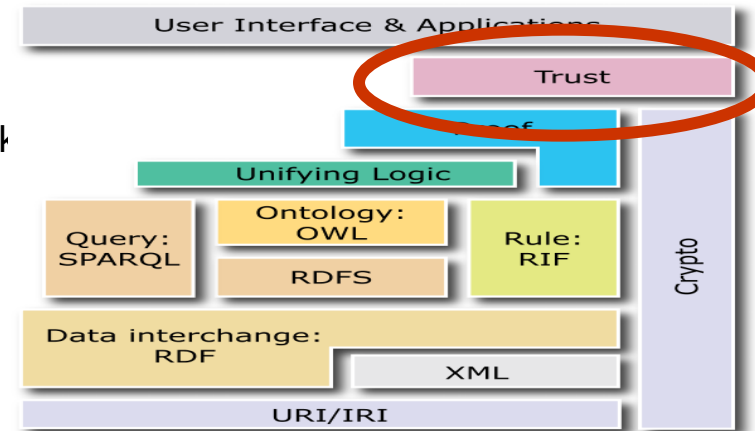


Web of Trust – Erste Ideen

- Behauptungen können verifiziert werden, wenn es aus anderen (vertrauenswürdigen) Quellen (im Internet) Belege dafür gibt

- Beispiel 1

- Wenn wir wissen:
 - `unterorganisation(CSW, Institut für Informatik)`
 - `unterorganisation(Fakultät für Mathematik und Informatik FUB)`
- Und die Web Site sagt:
 - `arbeitet_fuer(Adrian Paschke, CSW)`
- dann können wir ableiten:
 - `arbeitet_fuer(Adrian Paschke, Fakultät für Informatik)`
 - `arbeitet_fuer(Adrian Paschke, FUB)`



- Beispiel 2

- Sie glauben jede Aussage aus verlässlichen Zeitungen
 - `believe(x) :- claims(src, x) ^ reliableNewspaper(src)`
- Sie legen fest, dass die Süddeutsche Zeitung ist eine verlässliche Zeitung
 - `isa(http://www.bz-berlin.de, reliableNewspaper)`

Semantic Web Tools / User Interfaces

Triple Stores

RDFStore, AllegroGraph, Tucana
RDF Gateway, Mulgara, SPASQL
Jena's SDB, D2R Server, SOR
Virtuoso, Oracle11g
Sesame, OWLIM, Tallis Platform

...

Reasoners / Rule Engines

Prova, OO jDrew
Pellet, RacerPro, KAON2, FaCT++
Ontobroker, Ontotext
SHER, Oracle 11g, AllegroGraph

...

Converters

flickurl, TopBraid Composer
GRDDL, Triplr, jpeg2rdf

...

Search Engines

Falcon, Sindice, Swoogle

...

Middleware

Rule Responder, IODT, Open Anzo, DartGrid
Ontology Works, Ontoprise
Profium Semantic Information Router
Software AG's EII
Thetus Publisher, Asio, SDS

...

Semantic Web Browsers

Disco, Tabulator, Zitgist, OpenLink Viewer

...

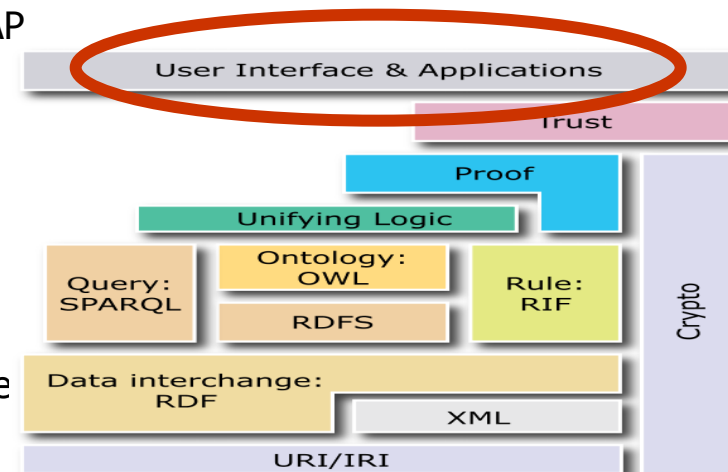
Development Tools

SemanticWorks, Protégé
Jena, Redland, RDFLib, RAP
Sesame, SWI-Prolog,
Prova Rule Manager
TopBraid Composer
DOMÉ

...

Semantic Wiki systems

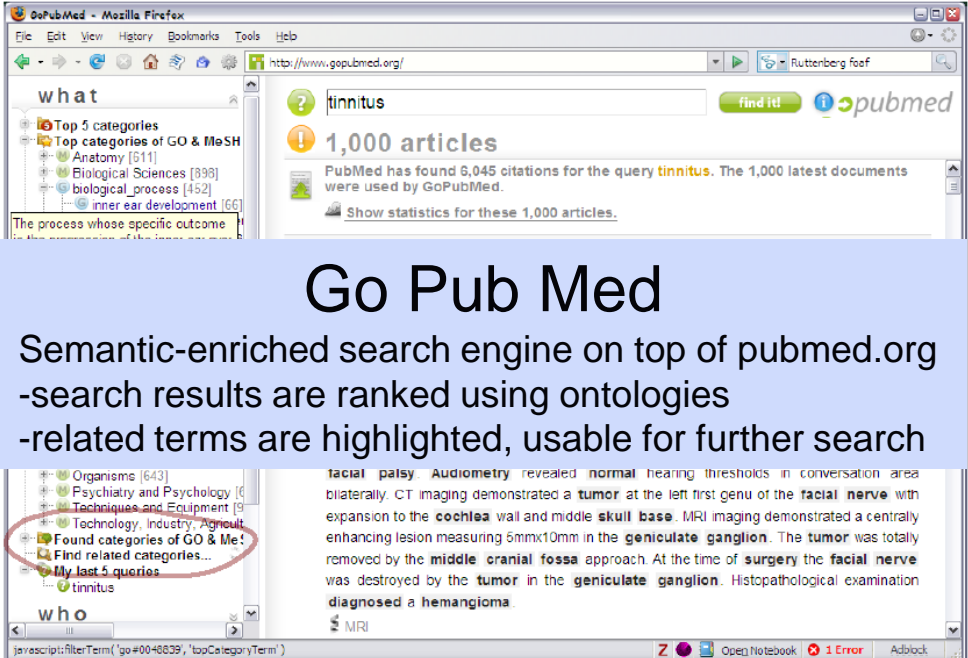
Semantic Media Wiki,
Platypus, Visual knowledge



Use Cases / Anwendungsgebiete

- Semantic-enriched Search
- Content management
- Knowledge management
- Business intelligence
- Collaborative user interfaces
- Sensor-based services
- Linking virtual communities
- Grid infrastructure
- Multimedia data management
- **Semantic Web Services**
- Etc. see e.g. SWEO's use case collection

<http://www.w3.org/2001/sw/sweo/public/UseCases/>



Go Pub Med

Semantic-enriched search engine on top of pubmed.org
-search results are ranked using ontologies
-related terms are highlighted, usable for further search

facial palsy. Audiometry revealed normal hearing thresholds in conversation area bilaterally. CT imaging demonstrated a tumor at the left first genu of the facial nerve with expansion to the cochlea wall and middle skull base. MRI imaging demonstrated a centrally enhancing lesion measuring 6mmx10mm in the geniculate ganglion. The tumor was totally removed by the middle cranial fossa approach. At the time of surgery the facial nerve was destroyed by the tumor in the geniculate ganglion. Histopathological examination diagnosed a hemangioma.

Links

- RuleML <http://www.ruleml.org/>
- Reaction RuleML
<http://ibis.in.tum.de/research/ReactionRuleML/>
- W3C RIF
http://www.w3.org/2005/rules/wiki/RIF_Working_Group
- OMG PRR <http://www.omg.org/docs/dtc/07-11-04.pdf>
- OMG SBVR <http://www.omg.org/spec/SBVR/1.0/>
- Rule Responder <http://responder.ruleml.org/>
- Prova
 - <http://prova.ws/>
 - Paschke, A.: Rule-Based Service Level Agreements - Knowledge Representation for Automated e-Contract, SLA and Policy Management, ISBN 978-3-88793-221-3, Idea Verlag GmbH, München, 2007.