Semantic Web Application Models

Invited Talk at the
ER 2003 - 22nd International Conference on Conceptual Modeling, Chicago, Illinois
13-16 October, 2003

Erich Neuhold
Claudia Niederée
Michael Fuchs
Fraunhofer Institute IPSI
Darmstadt Germany
http://ipsi.fhg.de

Content

- Trends for the next generation of the Web
  - The Semantic Web
  - Web Services
- Challenges for Web applications in the Next Generation of the WWW
- Semantic Web Application Models and related Standards
- Operationalization: A Semantic Web Application Development Framework
- Open Issues

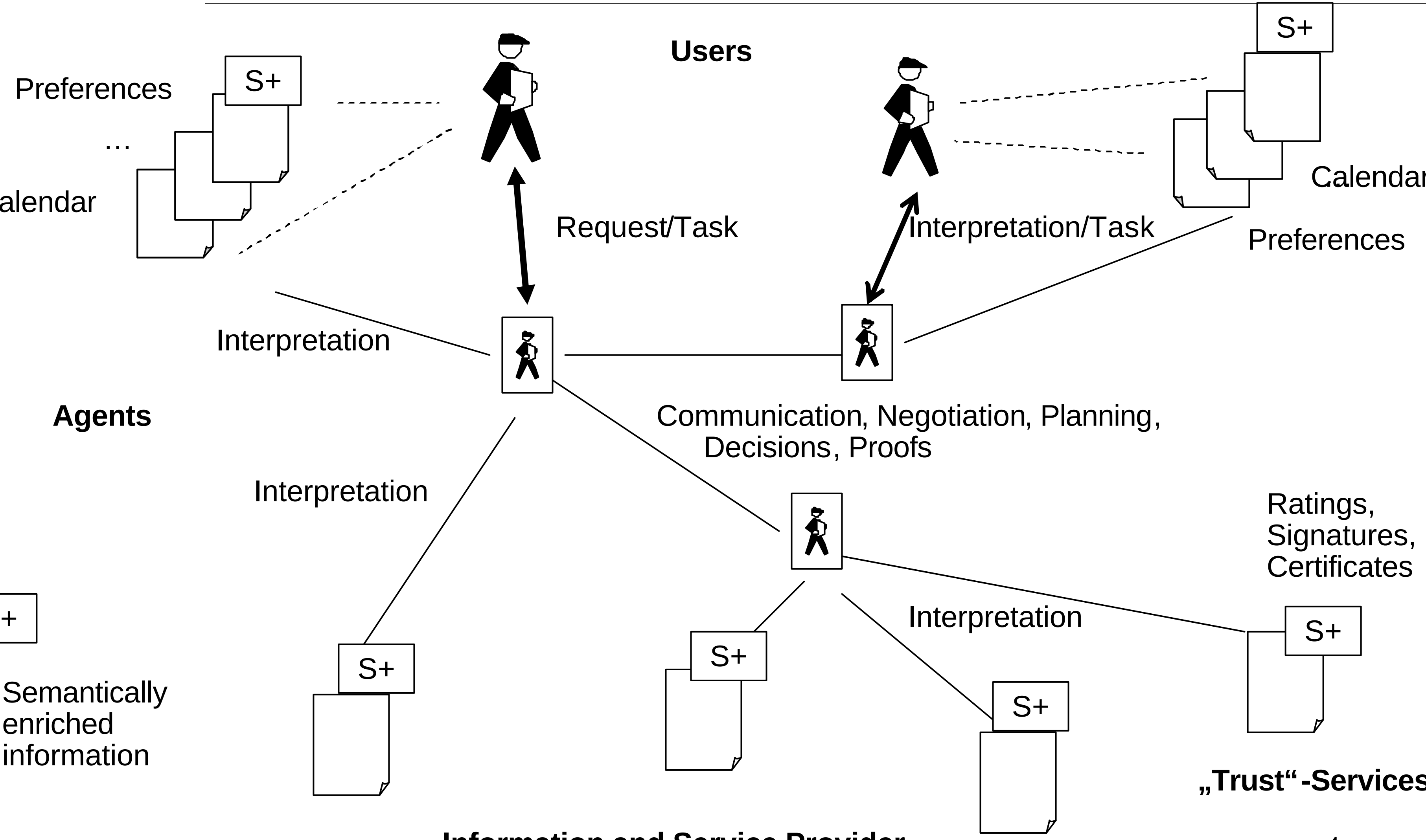
The Next Generation of the Web

Important trends for the next Generation of the World Wide Web

- The Web as “Programming Interface”:
Web Service Paradigm
- Semantic Enrichment for improved information and service mediation: **Semantic Web**

"*The **Semantic Web** is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.*"
Tim Berners-Lee, James Hendler, Ora Lassila, The Semantic Web, Scientific American, May 2001

Semantic Web - Vision



Users

Preferences

…

alendar

Request/Task

Interpretation/Task

Calendar

Preferences

Interpretation

Agents

Communication, Negotiation, Planning,
Decisions, Proofs

Interpretation

Ratings,
Signatures,
Certificates

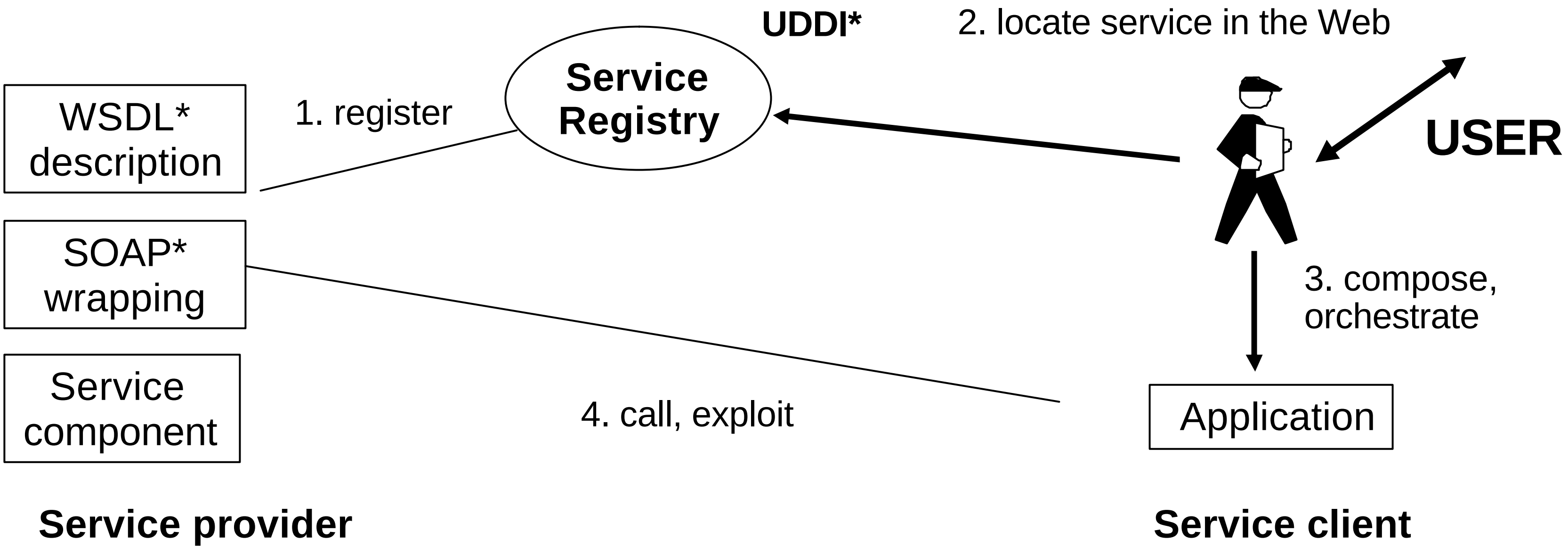
Semantically
enriched
information

Interpretation

„Trust“-Services

Information and Service Provider

The Web Services Architecture



UDDI*

2. locate service in the Web

WSDL* description

1. register

Service Registry

USER

SOAP* wrapping

3. compose, orchestrate

Service component

4. call, exploit

Application

Service provider

Service client

Web Services support the flexible and dynamic configuration of IT and service infrastructures

***SOAP = Simple Object Access Protocol**
***WSDL = Web Service Description Language**
***UDDI = Universal Description, Discovery and Integrati**

Challenges for Web Applications

For effective operation in the Semantic Web, Web applications have to

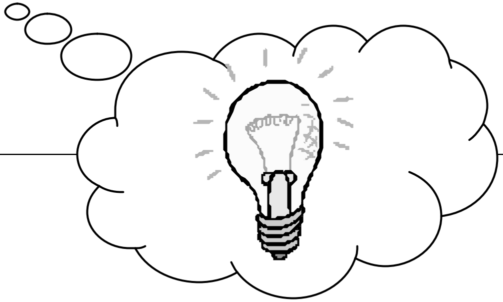
- move from a purely human user community of the WWW towards a mixed user community (**humans as well as software agents**);
- support information to enable **automatic interpretation** of delivered Web page content; (interlinking local data and content with globally defined interpretation schemes like **vocabularies and ontologies**);

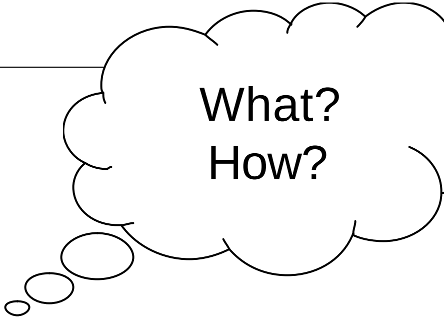
General Challenges for the Service Paradigm:

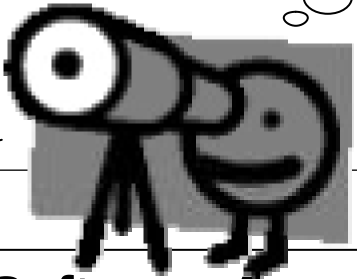
- Service Discovery
- Service Composition

Current Situation

Web
Client
(HTML)







Software Agent

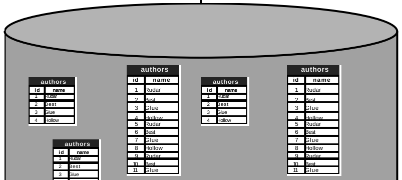
User Interface Layer

JSP*, ASP*, …

Application Layer

ODBC*, JDBC*, …

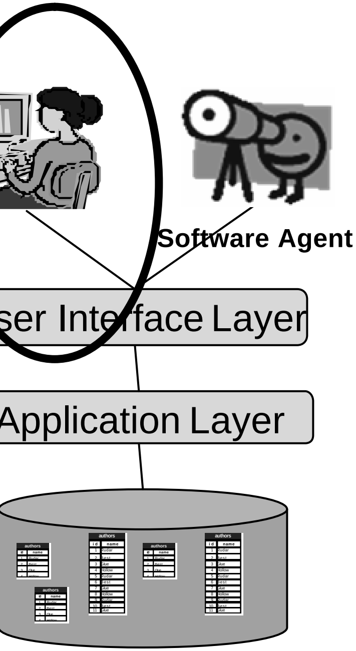
not model-based,
semi-structured document (HTML), but
without explicit Schemata



*JDBC/ODBC = Java/Open Database Connectivity
*JSP/ASP = Java/Active Sever Pages

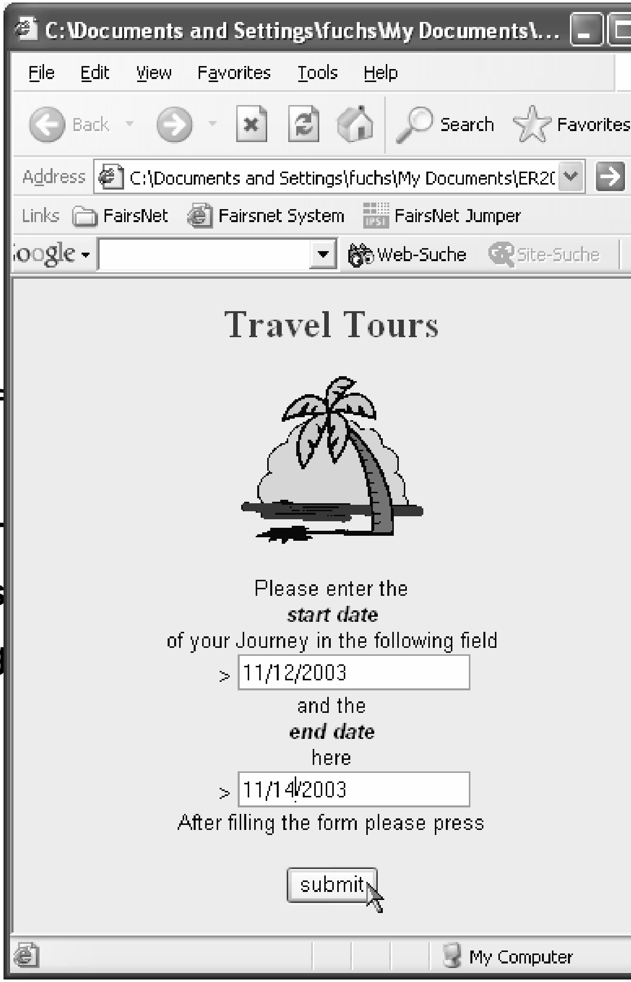
HTML-based UI Dialog Models - Shortcomings

- no (or little) client side validation
- restricted client side interaction
- restricted control of submitted data
- **strong bias** towards one user interface agent (**Web Browser**)
- static form composition
- HTML needs not to be "well formed" (**not parseable**)
- mixing layout and structure



HTML – Properties and Shortcomings

```
html>
BODY bgcolor=yellow>
<form action=followUpPage.cgi><p>
 <CENTER><h2><font color=red>Travel Tours</h2></font>
 <P><img src="palm.gif"></p>
 <font face=Arial size="2">
  Please enter the <br><i><b>start date</i></b> <br>of
  Journey in the following field
  <br>&nbsp;&nbsp;&nbsp; > <input type=text name='star
  and the <br><i><b>end date</i></b><br> here <br>&nbs
  > <input type="input" name='end'> <BR> After filling
   form please press <br>
   <p><input type='submit' value="submit"></P>
</FORM>

/Body>
/Html>
```



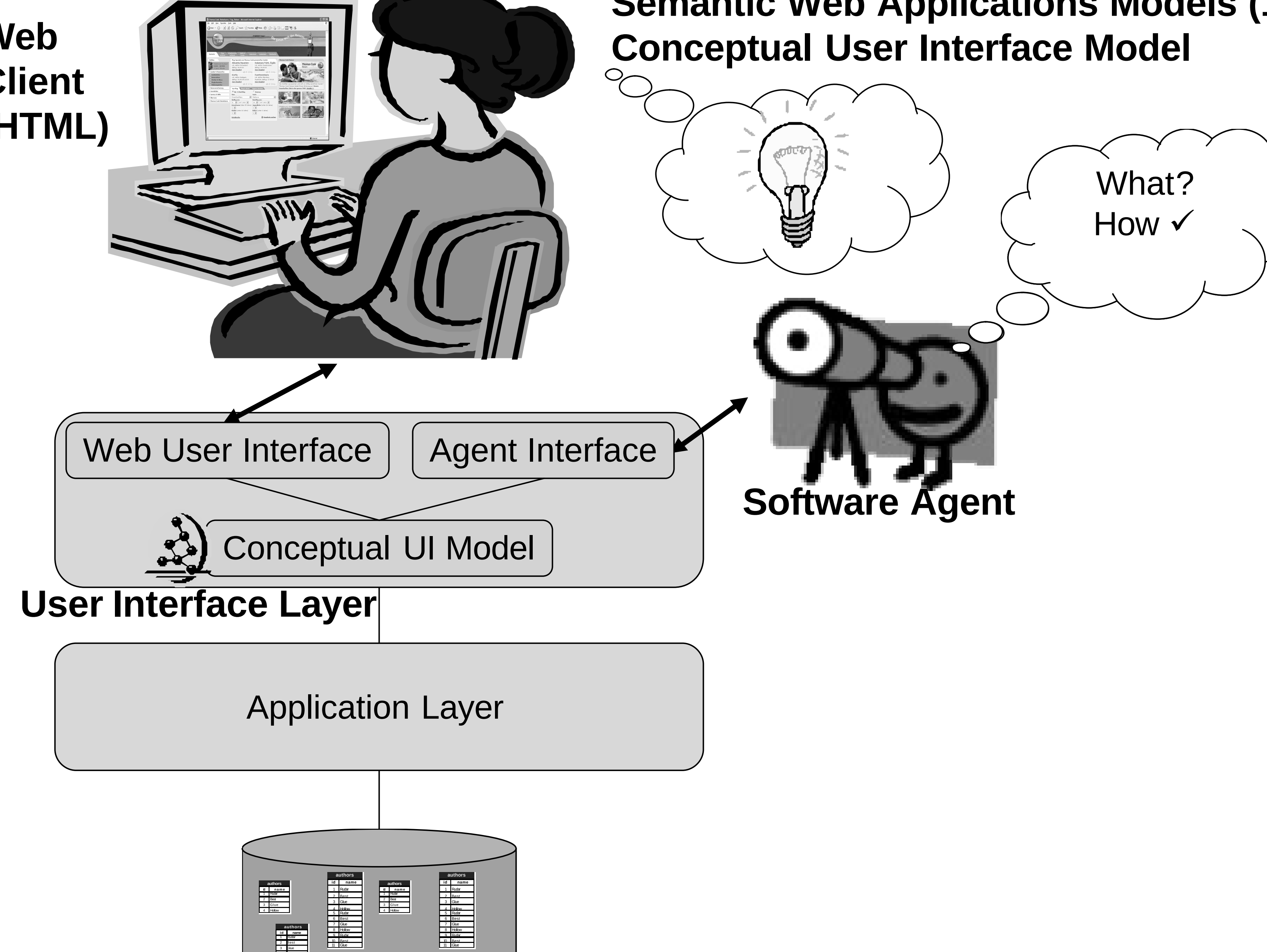
Travel Tours

Please enter the
start date
of your Journey in the following field
> 11/12/2003
and the
end date
here
> 11/14/2003
After filling the form please press

submit

XML Family ??

Semantic Web Applications Models (
Conceptual User Interface Model



Web
Client
(HTML)

What?
How ✓

Web User Interface

Agent Interface

Conceptual UI Model

User Interface Layer

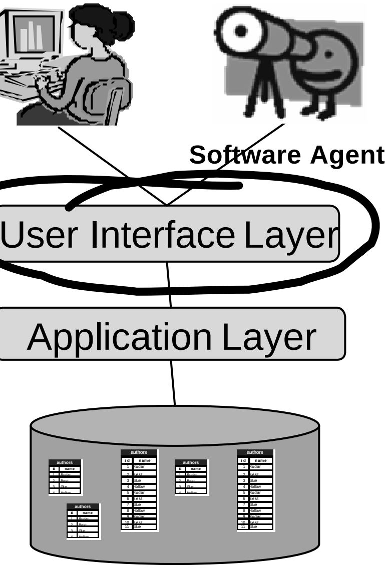
Software Agent

Application Layer

Requirements for Conceptual UI Model

Conceptual model for form-based User Interfaces requires

- design of **forms on an abstract level** → independence of specific user/agent interfaces
- rich set of UI component **types**
- more **powerful client-side** interaction model
- dynamic client side behavior



Operationalization

- processors for different user/agent interfaces
- comfortable editors for form authoring

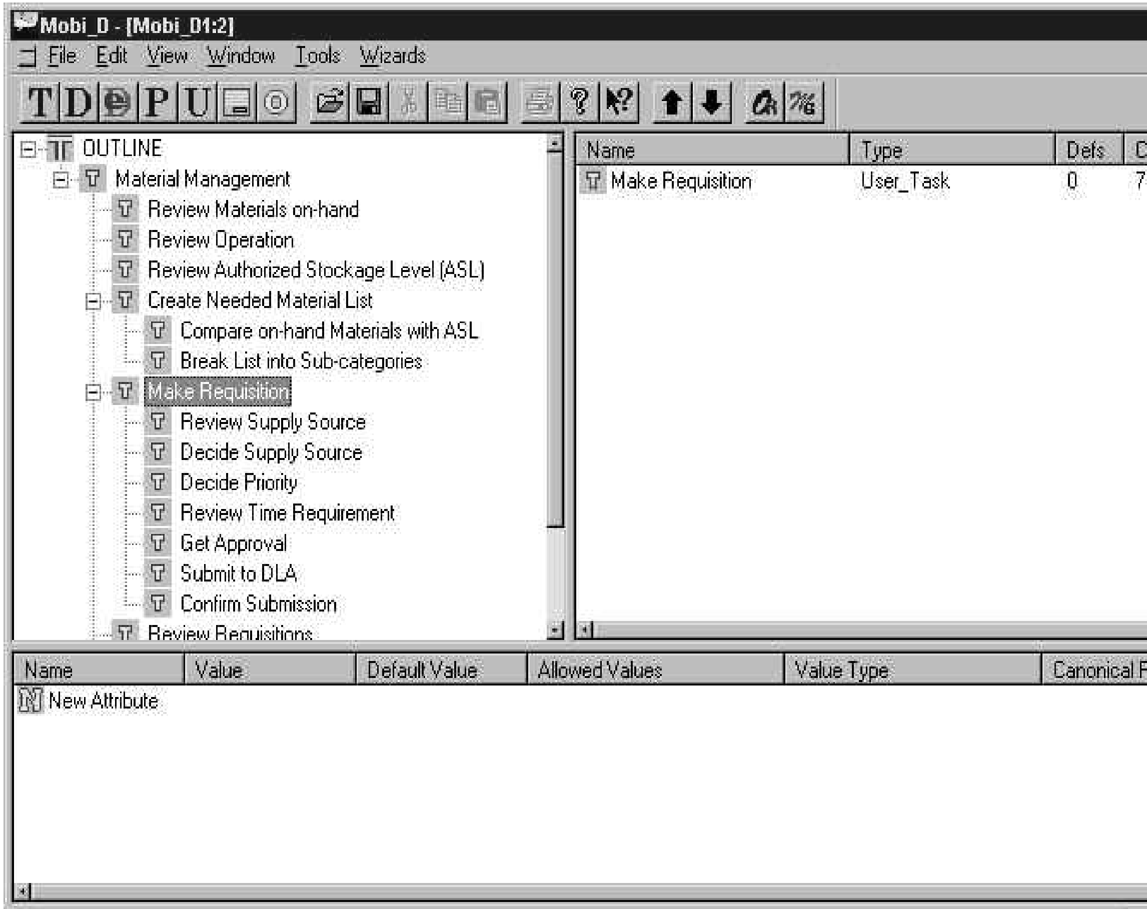
Use of Models for User Interface Creation

Mecano and MOBI-D (Model-based Interface Designer)

The Mecano Interface Models [Puerta 1997]

1. **User Model**
2. **User-Task Model**
3. **Domain Model**
4. **Presentation Model**
5. **Dialog Model**
6. **Design Model**

Uses object oriented modeling language: **MIMIC***



- OUTLINE
  - Material Management
    - Review Materials on-hand
    - Review Operation
    - Review Authorized Stockage Level (ASL)
    - Create Needed Material List
      - Compare on-hand Materials with ASL
      - Break List into Sub-categories
    - Make Requisition
      - Review Supply Source
      - Decide Supply Source
      - Decide Priority
      - Review Time Requirement
      - Get Approval
      - Submit to DLA
      - Confirm Submission
    - Review Requisitions

Name	Type	Defs	C
Make Requisition	User_Task	0	7

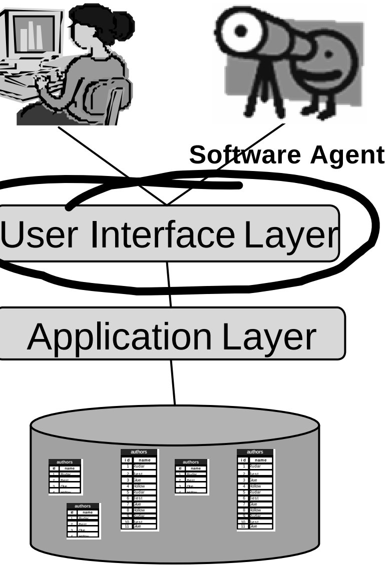
Name	Value	Default Value	Allowed Values	Value Type	Canonical F
New Attribute					

Mecano Interface Modeling Language

Conceptual UI Model - XForms

XForms - the next generation of form-based user interfaces:

- design of forms on a more abstract level → independence of specific user/agent interfaces ✓
- rich set of UI component types (XSLT, XML acc.) ✓
- more powerful client-side interaction model ✓
- **dynamic client side behavior** (e.g. add new fields or exchange form blocks while using the form) ✓



Operationalization (not supported by XForms)

- processors for different user interface agents
- comfortable editors for form authoring

XForms Example –SCHOLNET Project (1)

Instance of

Data Model

```
<SearchForm>
                    ⋮

<fields>
  <field name="dc:creator" type="string">
    <rel-ops selected="equal">
      <rel-op name="equal" symbol="" position="infix"/>
                    ⋮
    </rel-ops>
                    ⋮
 </fields>
                    ⋮
</SearchForm>
```

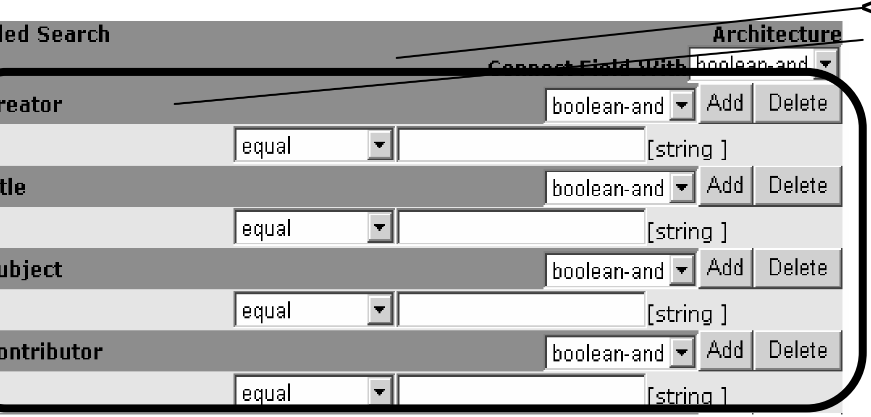
Conceptual

User Interface Model

```
<xforms:repeat nodeset="SearchForm/fields/field"/>
 <xforms:select1 ref="field-operators/@selected">
  <xforms:itemset nodeset="field-operators/field-operator">
   <xforms:caption ref="@name"/><xforms:value ref="@name
  </xforms:itemset>
 </xforms:select1>
 <xforms:button value="Add" onclick="new:rptName"/>
 <xforms:button value="Delete" onclick="delete:rptName"/>
 <xforms:group  nodeset="rel-ops"/>
  <xforms:select1 ref="@selected" appearance="min">
   <xforms:itemset nodeset="rel-op">
    <xforms:caption ref="@name"/><xforms:value ref="@name"/>
   </xforms:itemset>
  </xforms:select1>
  <xforms:input ref="value"/>[<xforms:output ref="../@type"/>]
 </xforms:group>
```

XForms Example –SCHOLNET Project

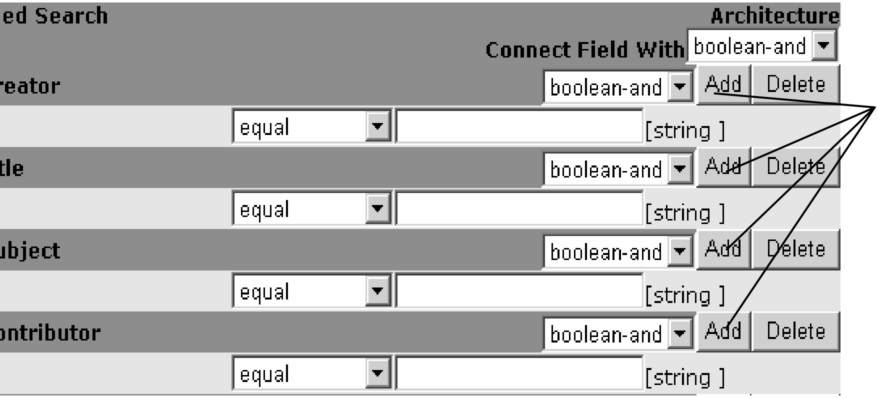
```
<SearchForm>
                    ⋮

<fields>
  <field name="dc:creator" type="string">
    <rel-ops selected="equal">
      <rel-op name="equal" symbol="" position="infix"/>
                    ⋮
    </rel-ops>
                    ⋮
 </fields>
                    ⋮
</SearchForm>
<xforms:repeat nodeset="SearchForm/fields/field"/>
 <xforms:select1 ref="field-operators/@selected">
  <xforms:itemset nodeset="field-operators/field-operator">
    <xforms:caption ref="@name"/><xforms:value ref="@name
  </xforms:itemset>
 </xforms:select1>
 <xforms:button value="Add" onclick="new:rptName"/>
 <xforms:button value="Delete" onclick="delete:rptName"/>
 <xforms:group  nodeset="rel-ops"/>
  <xforms:select1 ref="@selected" appearance="min">
    <xforms:itemset nodeset="rel-op">
      <xforms:caption ref="@name"/><xforms:value ref="@name"/>
    </xforms:itemset>
  </xforms:select1>
  <xforms:input ref="value"/>[<xforms:output ref="../@type"/>]
 </xforms:group>
```



led Search Architecture
Connect Field With boolean-and
reator boolean-and Add Delete
equal [string]
tle boolean-and Add Delete
equal [string]
ubject boolean-and Add Delete
equal [string]
ontributor boolean-and Add Delete
equal [string]

XForms Example – SCHOLNET Project

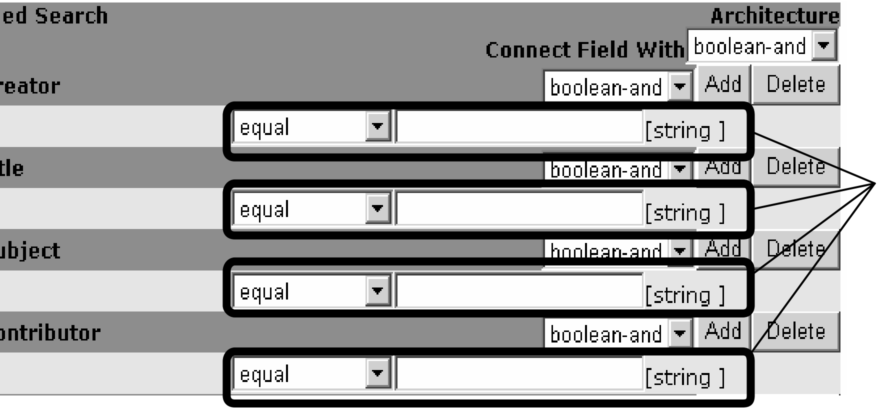
```
<SearchForm>
                    ⋮
<fields>
   <field name="dc:creator" type="string">
     <rel-ops selected="equal">
       <rel-op name="equal" symbol="" position="infix"/>
                    ⋮
     </rel-ops>
                    ⋮
 </fields>
                    ⋮
</SearchForm>
<xforms:repeat nodeset="SearchForm/fields/field"/>
 <xforms:select1 ref="field-operators/@selected">
   <xforms:itemset nodeset="field-operators/field-operator">
     <xforms:caption ref="@name"/><xforms:value ref="@name"/>
   </xforms:itemset>
 </xforms:select1>
 <xforms:button value="Add" onclick="new:rptName"/>
 <xforms:button value="Delete" onclick="delete:rptName"/>
 <xforms:group  nodeset="rel-ops"/>
   <xforms:select1 ref="@selected" appearance="min">
     <xforms:itemset nodeset="rel-op">
       <xforms:caption ref="@name"/><xforms:value ref="@name"/>
     </xforms:itemset>
   </xforms:select1>
   <xforms:input ref="value"/>[<xforms:output ref="../@type"/>]
 </xforms:group>
```



ed Search   Architecture
Connect Field With boolean-and
reator   boolean-and Add Delete
equal [string]
tle   boolean-and Add Delete
equal [string]
ubject   boolean-and Add Delete
equal [string]
ontributor   boolean-and Add Delete
equal [string]

XForms Example – SCHOLNET Project

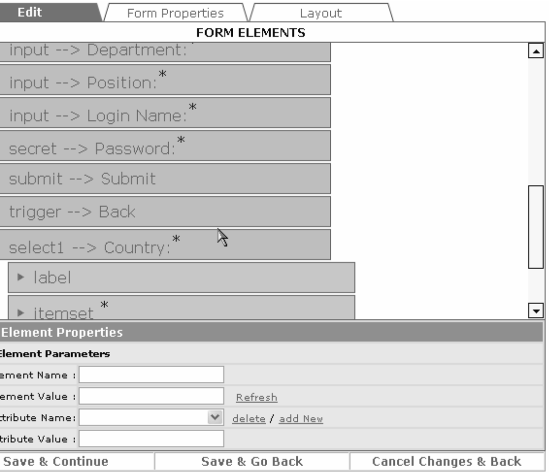
```
<SearchForm>
                    ⋮
<fields>
  <field name="dc:creator" type="string">
   <rel-ops selected="equal">
    <rel-op name="equal" symbol="" position="infix"/>
                    ⋮
   </rel-ops>
                    ⋮
 </fields>
                    ⋮
</SearchForm>
<xforms:repeat nodeset="SearchForm/fields/field"/>
 <xforms:select1 ref="field-operators/@selected">
  <xforms:itemset nodeset="field-operators/field-operator">
   <xforms:caption ref="@name"/><xforms:value ref="@name"/>
  </xforms:itemset>
 </xforms:select1>
 <xforms:button value="Add" onclick="new:rptName"/>
 <xforms:button value="Delete" onclick="delete:rptName"/>
 <xforms:group  nodeset="rel-ops"/>
  <xforms:select1 ref="@selected" appearance="min">
   <xforms:itemset nodeset="rel-op">
    <xforms:caption ref="@name"/><xforms:value ref="@nam
   </xforms:itemset>
  </xforms:select1>
  <xforms:input ref="value"/>[<xforms:output ref="../@type"/>
 </xforms:group>
```

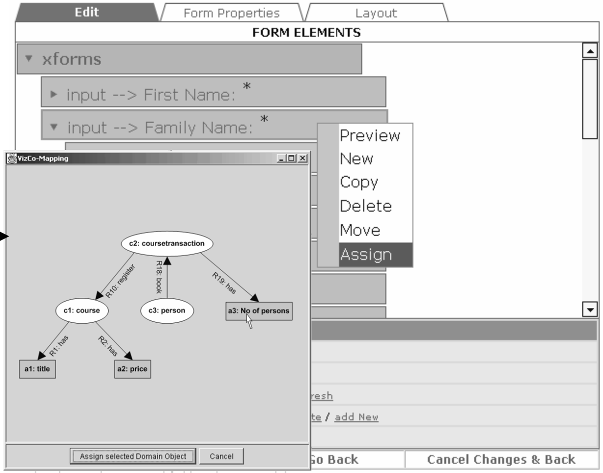


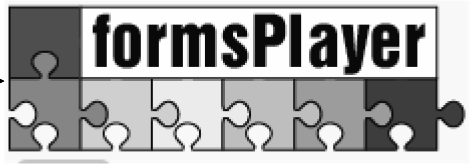
ed Search
Architecture
Connect Field With boolean-and
reator boolean-and Add Delete
equal [string]
tle boolean-and Add Delete
equal [string]
ubject boolean-and Add Delete
equal [string]
ontributor boolean-and Add Delete
equal [string]

How to use XForms?

Design time | Run time







Edit XForms Model

(Form Dialog Manager)

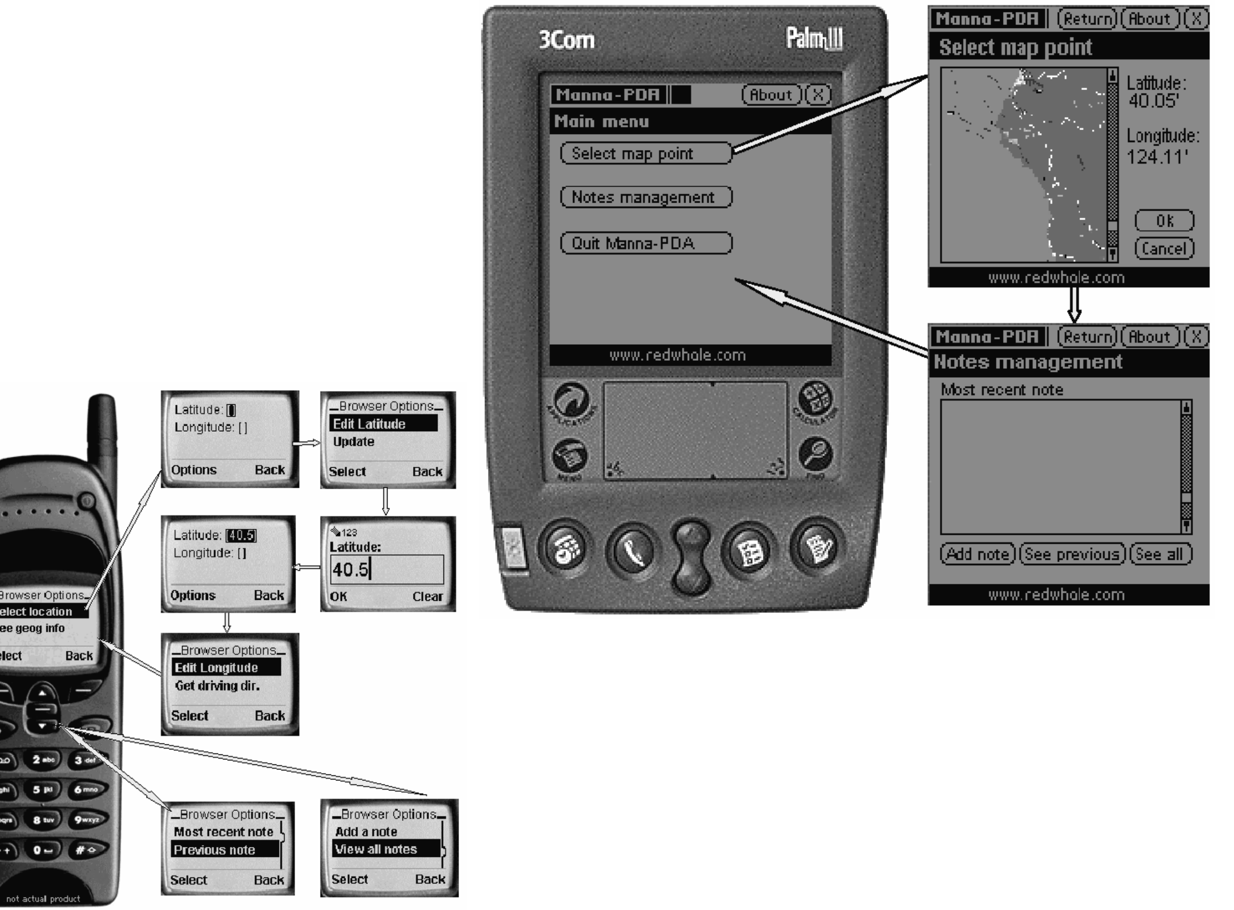
Mapping of XForms Model
⇔
Data Model
(VizCo *)

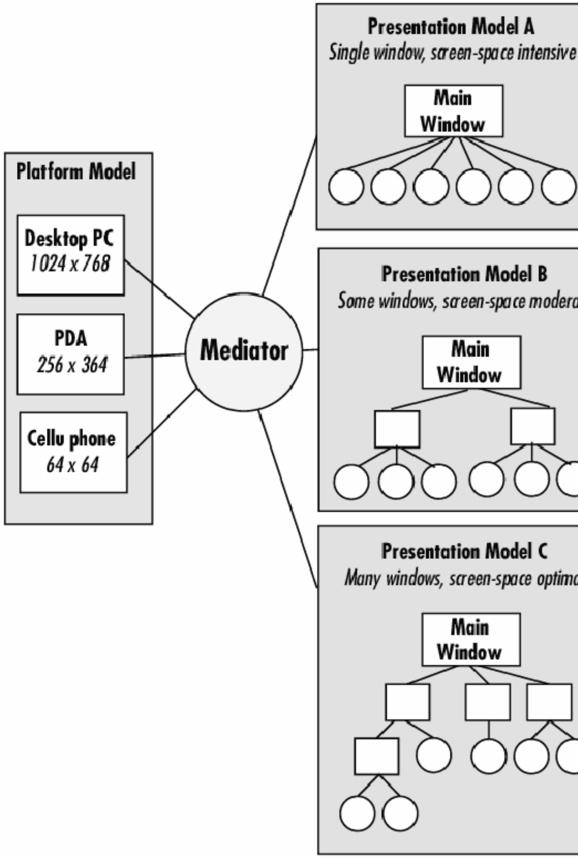
Transform XForm Model
⇔
Client Interface

FormsPLayer © x-por

*Fuchs, Niederée, Hemmje, Neuhold,
“Supporting Model-based Construction of Semantic-enabled Web Applications ”,
to appear in Proceedings of the 4th International Conference on Web Information Systems Engineering (WIS
Rome, December 2003

Use of Models for User Interface Generation



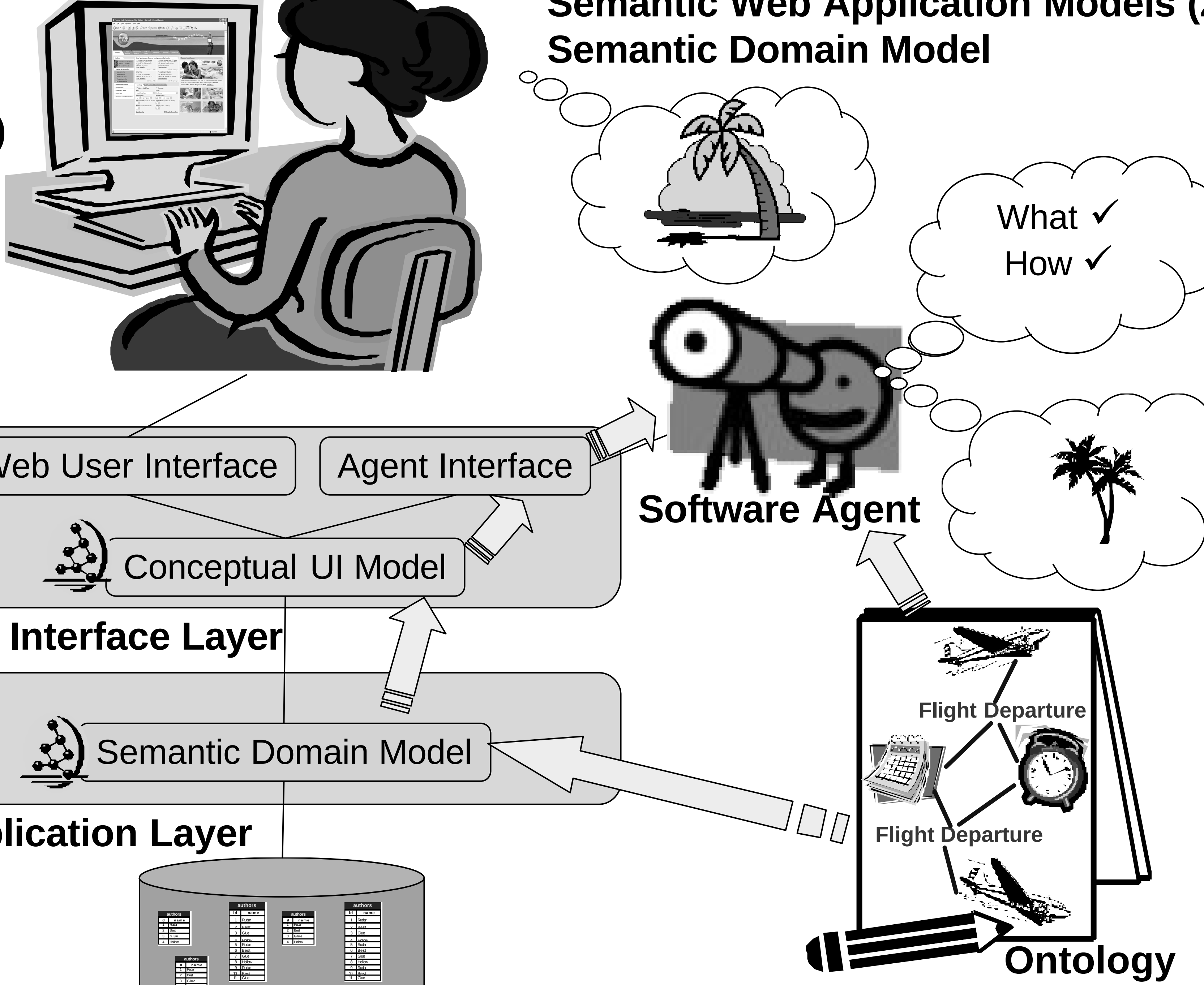


J. Eisenstein, J. Vanderdonckt, A. Puerta -- MIMIC based!

Applying Model-Based Techniques to the Development of UIs for Mobile Computers (2001)

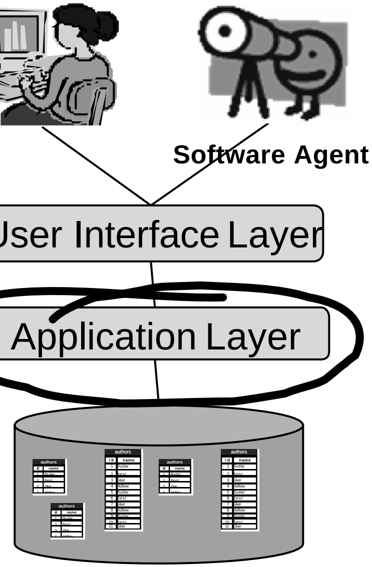
Semantic Web Application Models (2
Semantic Domain Model

Web
Client
(HTML)



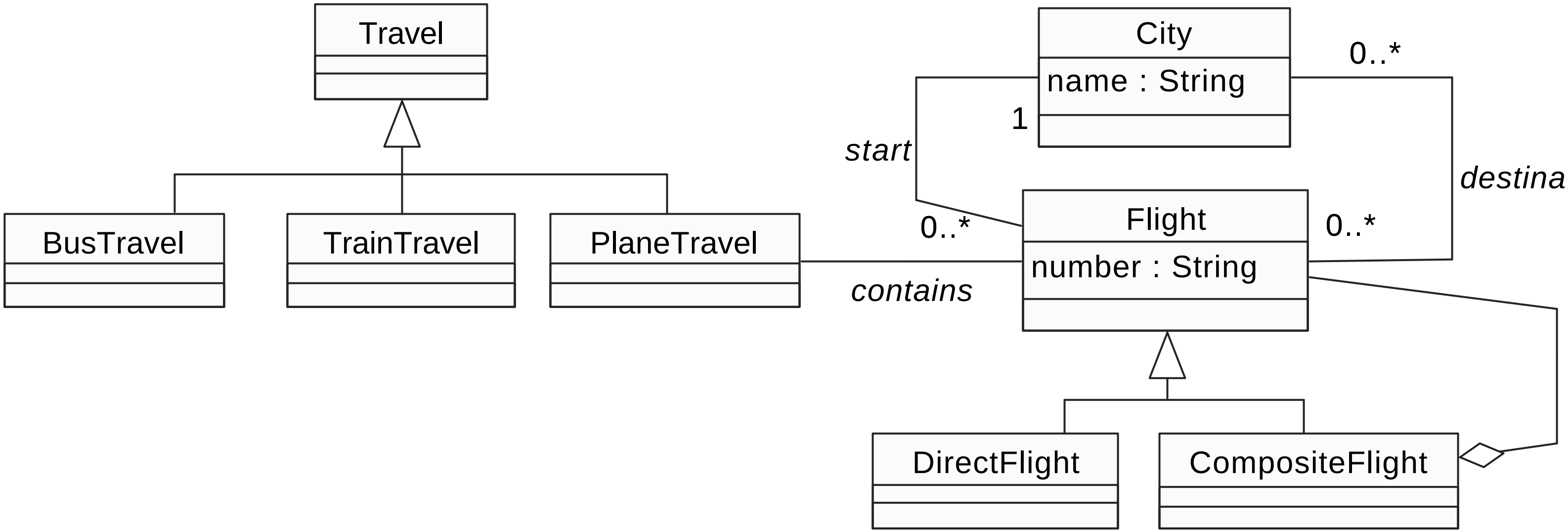
Requirements for Semantic Domain Model

- Support of modeling primitives (obj., rel., prop.)
- **Flexible definition of relationships to global conceptual models (Global Ontologies)**
- Representation of **concept hierarchies**
- Systematic support of data types
- Representation of additional domain knowledge and **constraints**
- Consistent support of domain/ontology evolution



Conceptual Modeling in Software Design

Example: Unified Modeling Language (**UML***)

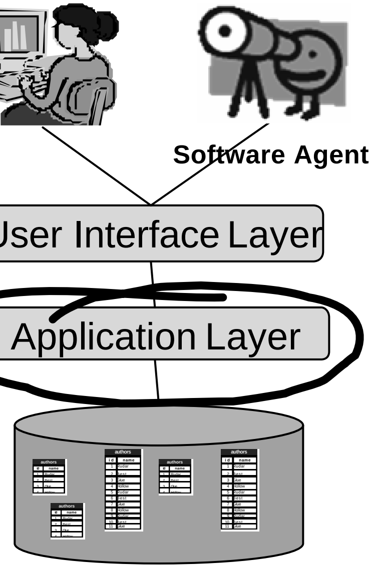


***UML Specification, Object Managent Group (OMG),**

see http://www.omg.org/technology/documents/formal/uml.htm

Fulfillment of Requirements - UML

- Support of modeling primitives
  - Definition of entities (class centric approach)
  - Definition of relationships (association, aggregation, composition)
  - Properties of relationships (multiplicities, association + role names, association classes)
- Flexible definition of relationships to global conceptual models
  - not supported
- Representation of concept hierarchies
  - subclass relationship
- Systematic support of data types
  - imported from DB Schema or Programming Language
- Representation of additional domain knowledge and constraints
  - in textual form
- Domain/Ontology evolution
  - not supported

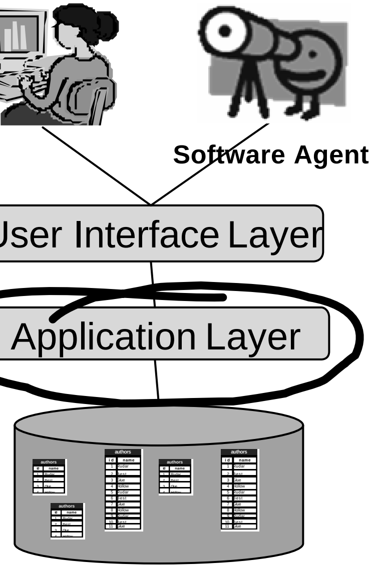


Main Focus is Software Design

Conceptual Modeling in the Semantic Web – RDF & Co

The **Resource Description Framework** (RDF + RDF Schema):

- Defines a framework for structuring and describing **resources** in the Semantic Web
- Enables the definition of **vocabularies** for the description of resources in an **application domain**;

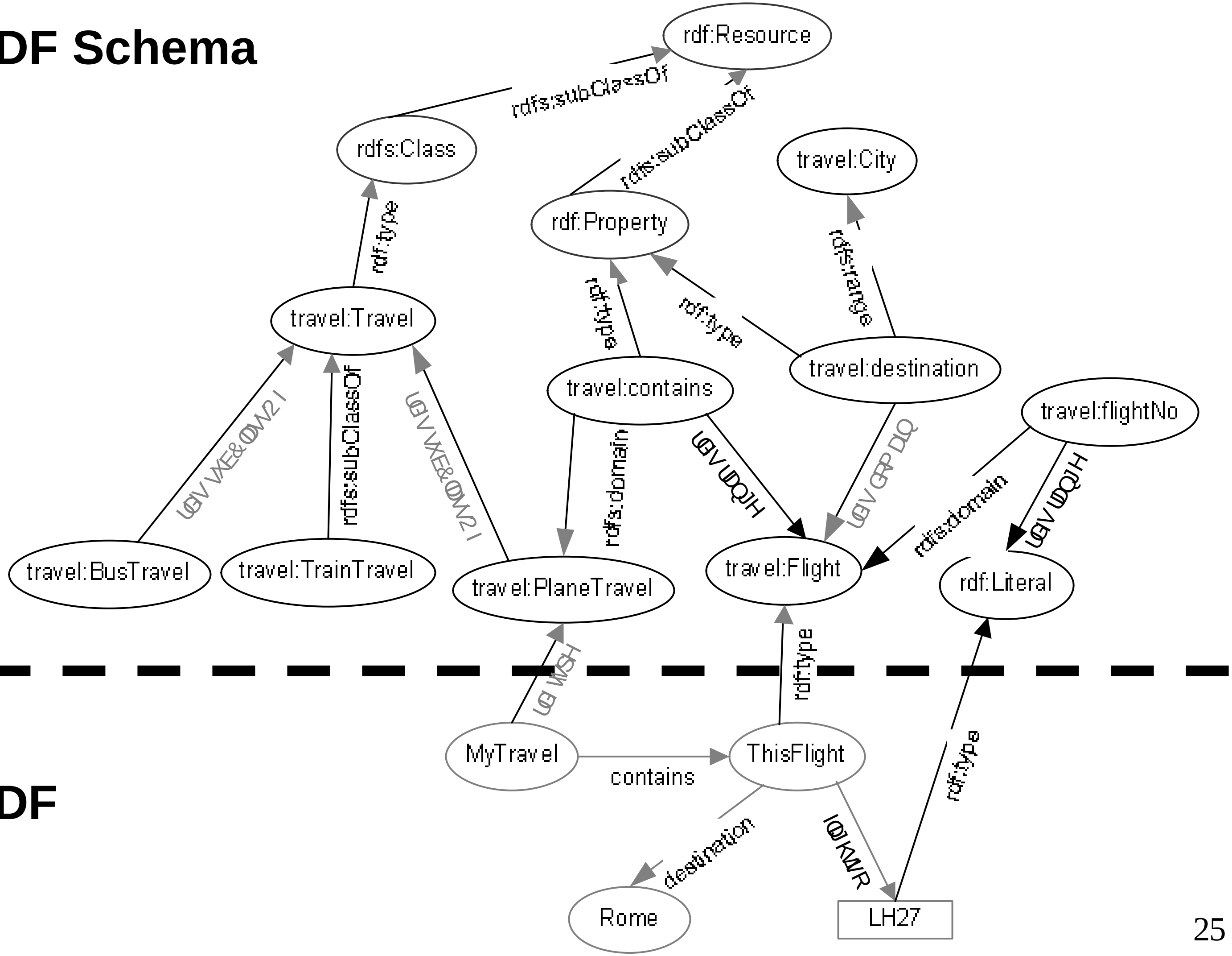


Goals:

- Extensibility, interoperability, and reuse of vocabularies;
- Improved support for interpretation of data by machines

Resource Description Framework – Example

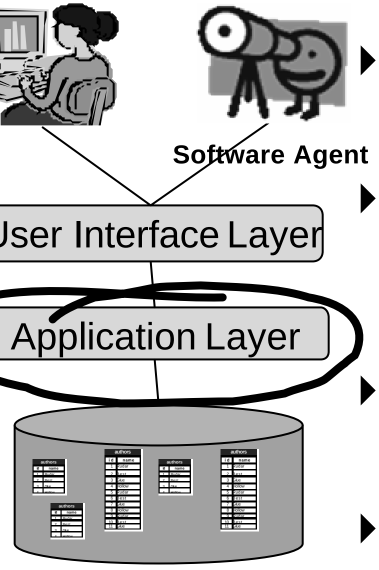
RDF Schema



RDF

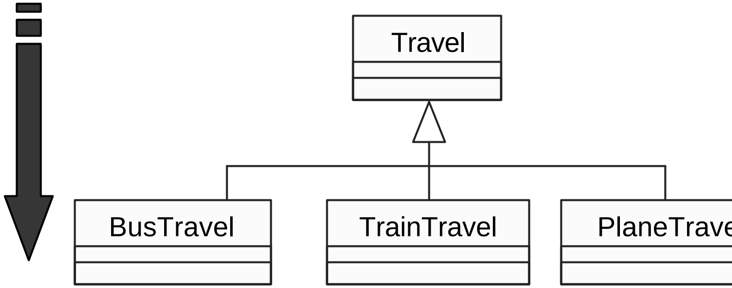
Fulfillment of Requirements - RDF Schema

- Support of **modeling primitives**
  - Definition of entities (classes + property centric approach)
  - Definition of relationships (via properties)
  - Properties of relationships (properties are also resources)
- Flexible definition of **relationships to global conceptual mode**
  - use of URI references, subclasses relationship (see next slide)
- Representation of **concept hierarchies**
  - yes, subclass relationship
- Systematic support of **data types**
  - only on the instance level (typed literal), use of XML Schema data types
- Representation of **additional** domain knowledge and constraints
  - no predefined concepts besides subclass + subproperty
- **Domain/Ontology** evolution
  - no



Relationships to global Conceptual Models - RDF

Example
“Different types of travel can be defined starting from the concept Travel defined in another global schemes”



“definition of new subclasses

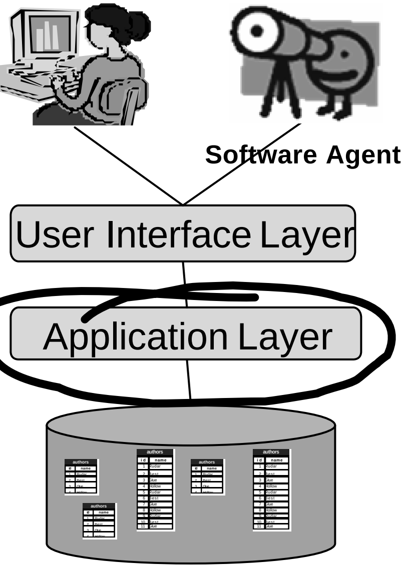
```
<rdfs:Class rdf:about="#busTravel">
     <rdfs:label>Bus Travel</rdfs:label>
     <rdfs:subClassOf rdf:resource="travel:Travel"/>
</rdfs:Class>
<rdfs:Class rdf:about="#planeTravel">
     <rdfs:label>Plane Travel</rdfs:label>
     <rdfs:subClassOf rdf:resource="travel:Travel"/>
</rdfs:Class>
…
```

Gobally
defined

What is still missing?

RDF contains only limited set of predefined concepts with defined semantics; concepts are missing for:

- the definition of **richer types of relationships** between conceptual models
- representation of **domain constraints** and further domain knowledge
- ontology/domain model **evolution** support



Defining Domain Ontologies -OWL

„An ontology is a specification of a conceptualization.” *

OWL (Web Ontology Language) is

- is a language for defining ontologies for the Semantic Web
- is currently developed by the W3C Web Ontology Group (Status: Working Draft) - successor of DAML + OIL
- is **building** upon RDF and RDF Schema
- enables the representation of **additional** domain knowledge (compared with RDF)

**T. R. Gruber. A translation approach to portable ontologies. Knowledge Acquisition, 5(2):199-220, 1993*

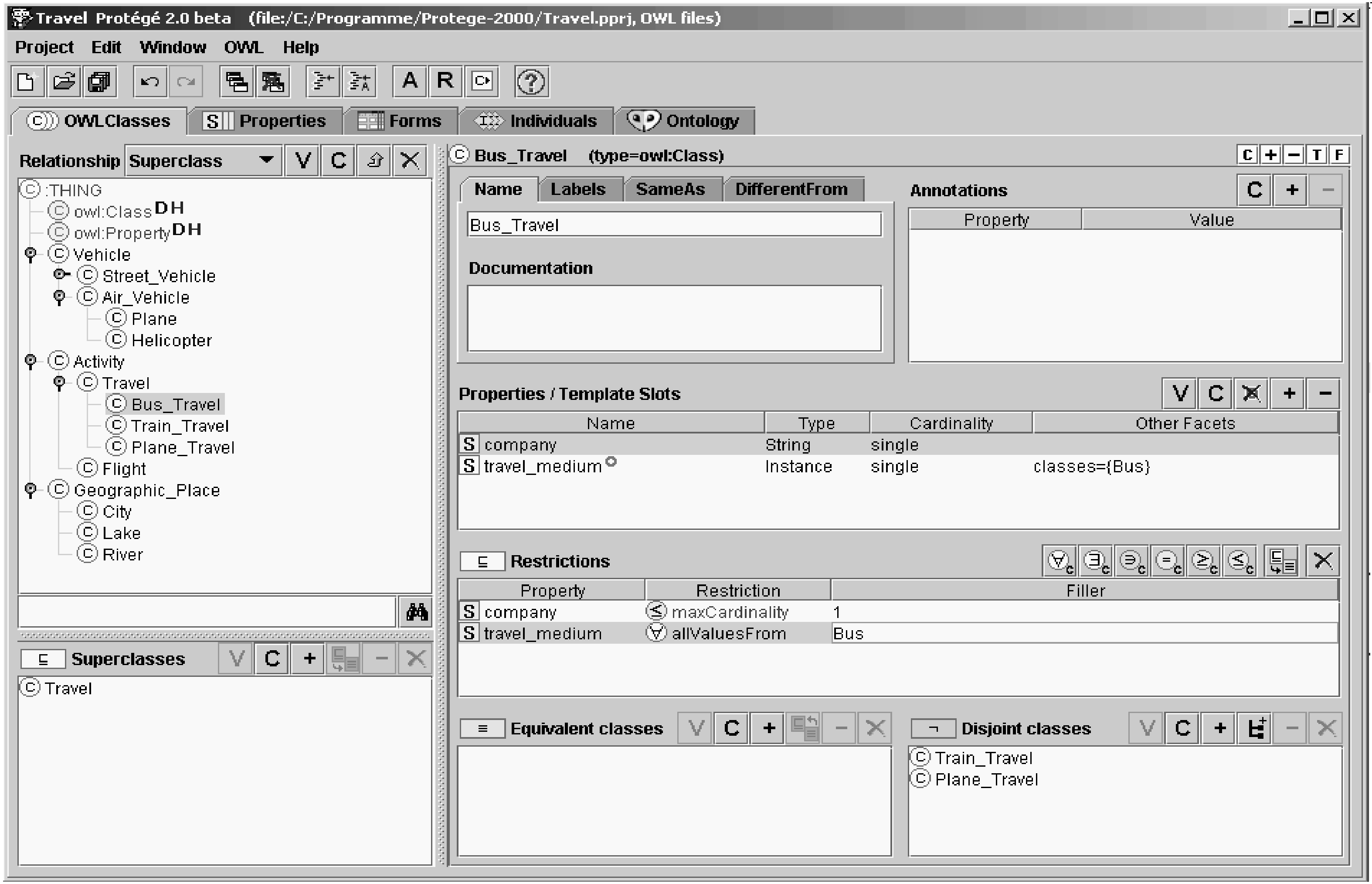
Role of OWL for Semantic Web Application Models

What are the **gains** of using **OWL** in Semantic Web Applications:

- Definition of ontologies that are used as **common reference** points for domain models;
- Specification of refined **relationships** between concepts in the **domain model and the ontology**;
- Formulation of **additional constraints** in the domain model;

→Advantage: predefined concepts **ease** automatic interpretation in **global distributed context**

Ontology Definition – Protégé (not OWL based)



Relationship: Superclass

```
:THING
  owl:Class DH
  owl:Property DH
  Vehicle
    Street_Vehicle
    Air_Vehicle
      Plane
      Helicopter
  Activity
    Travel
      Bus_Travel
      Train_Travel
      Plane_Travel
    Flight
  Geographic_Place
    City
    Lake
    River
```

Bus_Travel (type=owl:Class)

Name: Bus_Travel

Property	Value

Properties / Template Slots

Name	Type	Cardinality	Other Facets
S company	String	single	
S travel_medium	Instance	single	classes={Bus}

Restrictions

Property	Restriction	Filler
S company	≤ maxCardinality	1
S travel_medium	∀ allValuesFrom	Bus

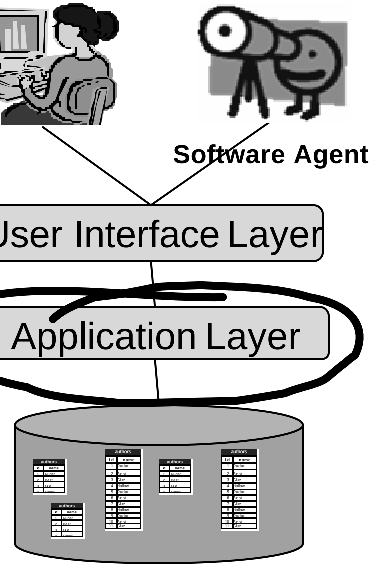
Superclasses: Travel

Disjoint classes: Train_Travel, Plane_Travel

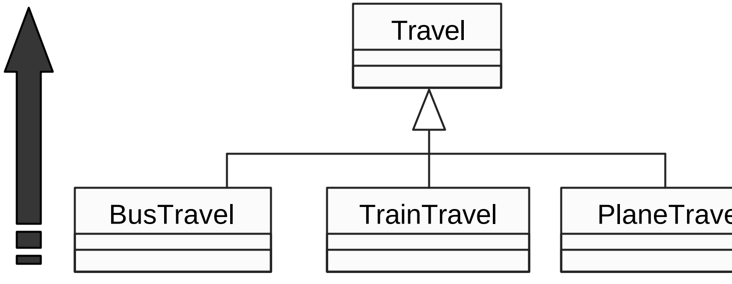
The Protégé Ontology Editor and Knowledge Acquisition System
see http://protege.stanford.edu/index.html

Fulfillment of Requirements - OWL

- Support of modeling primitives
  - Definition of entities (property centric approach like RDF)
  - Definition of relationships (via class properties)
  - Properties of relationships (cardinalities, transitivity, symmetry, ...)
- Flexible definition of relationships to global conceptual models
  - additional to RDF: disjoint, union, intersection, equivalent class, equivalent property
- Representation of concept hierarchies
  - yes like RDF + additional relationships, e.g. disjoint classes
- Systematic support of data types
  - on schema level (“data type properties”)
- Representation of additional domain knowledge and constraints
  - supported by predefined concepts, examples see next slides
- Domain/Ontology evolution
  - only basic support



Relationships to global Conceptual Models - OWL



Example

“A Travel is defined a bus travel, a plane or a train travel, assuming that this three types of travel are defined in other global schemes”

“definition of new superclass”

```
<owl:class rdf:about = “#Travel” >
   <owl:unionOf rdf:parseType= “Collection”>
      <owl:Class rdf:about = “airWays:Flight”
      <owl:Class rdf:about = “busWorld:Travel”
      <owl:Class rdf:about = “railway:TrainTravel”>
   </owl:unionOf>
</owl:class>
```

Representation of Domain Knowledge in OWL

Example

“A city can be either reached by a direct flight
or by connecting several flights”

```
<owl:transitiveProperty rdf:about = "#connect">
       <rdfs:domain rdf:about = "map:City">
       <rdfs:range rdf:about = "map:City">
</owl:transitiveProperty>

<owl:objectProperty rdf:about="travel:directFlight">
   <rdfs:subpropertyOf rdf:resource = "#connect">
</owl:objectProperty>
```

Additional Requirements for Semantic Web Applications

Mapping Tools between
and **Execution Environment**
for:

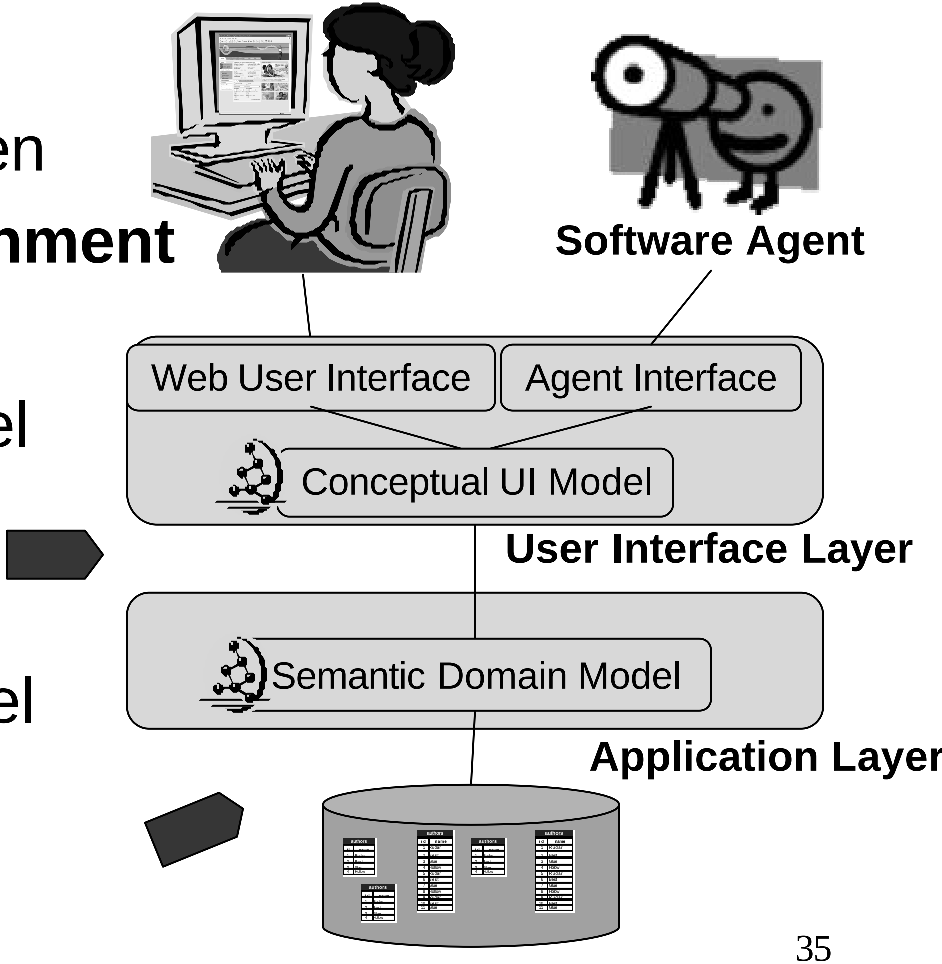
Conceptual UI Model

↕

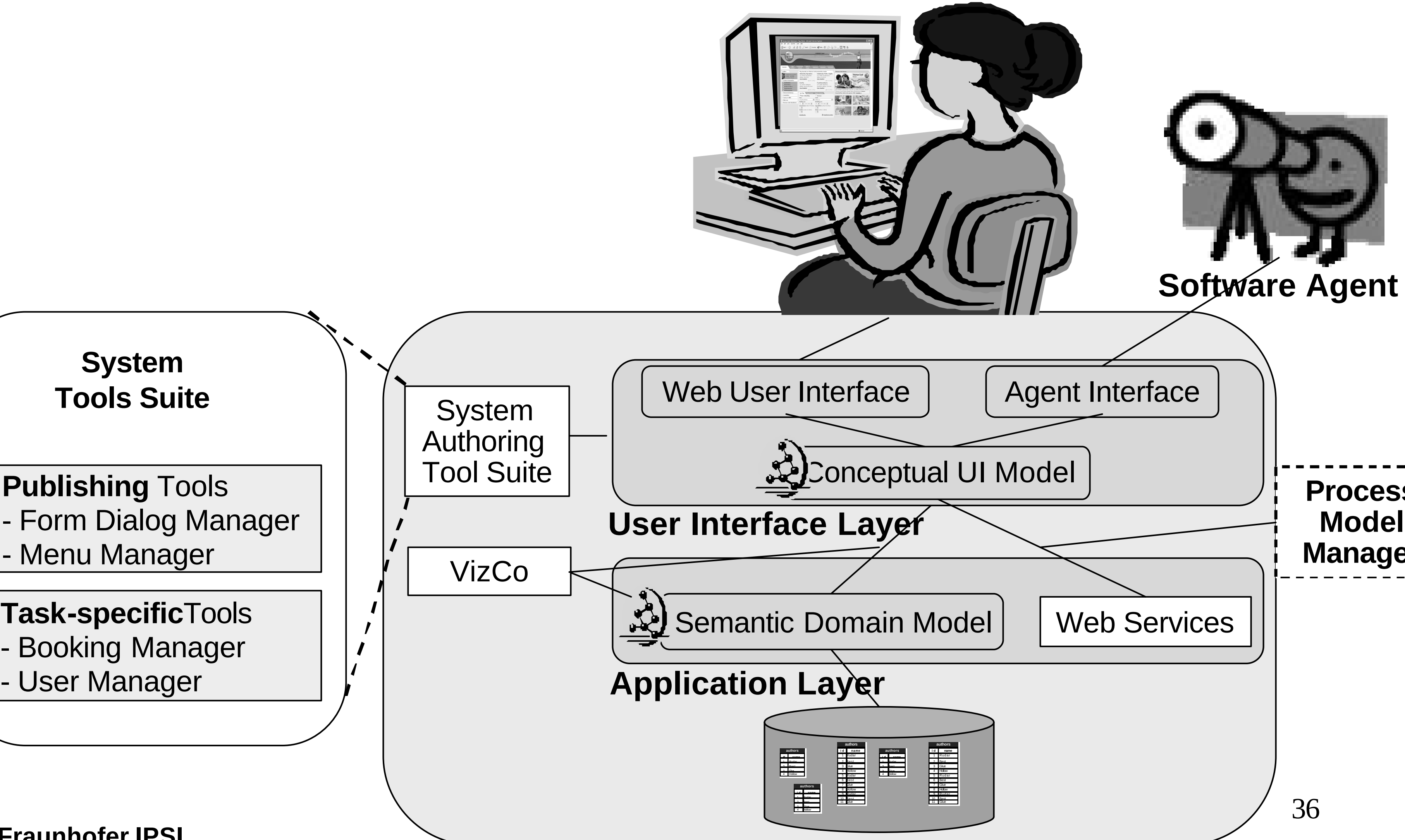
Semantic Domain Model

↕

Data Model



Operationalization: Model Authoring + Mapping Definition → FairsNet



Software Agent

System
Tools Suite

Publishing Tools
- Form Dialog Manager
- Menu Manager

Task-specific Tools
- Booking Manager
- User Manager

System
Authoring
Tool Suite

VizCo

Web User Interface

Agent Interface

Conceptual UI Model

User Interface Layer

Semantic Domain Model

Web Services

Application Layer

Proces
Model
Manage

Open Issues

- Development of **globally accepted (domain specific) Ontologies**
- Extended support for domain model and ontology **evolution**
- Systematic Handling of **multilinguality**
- Development of adequate processors for the transformation of conceptual UI models into **different agent-specific** UI formats
- **Standardization** and **Integration** of Process Models for WEB Services)

Thanks!