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

Agents

Preferences
Calendar

S+

Information and Service Provider

Users

Request/Task

Communication, Negotiation, Planning, Decisions, Proofs

Interpretation

Ratings, Signatures, Certificates

„Trust“-Services

Semantically enriched information

© Fraunhofer IPSI
The Web Services Architecture

1. register
2. locate service in the Web
3. compose, orchestrate
4. call, exploit

WSDL* description
SOAP* wrapping
Service component

Service Registry

Service provider
Service client

UDDI*
Application

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 Integration
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

User Interface Layer

JSP*, ASP*, ...

Application Layer

ODBC*, JDBC*, ...

Software Agent

What? How?

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

*JDBC/ODBC = Java/Open Database Connectivity
*JSP/ASP = Java/Active Server 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</font></h2></center>
        <p><img src="palm.gif"></p>
        <font face=Arial size="2">
          Please enter the <i>bstart date</i> of your Journey in the following field
          and the <i>end date</i> here
          <br>&nbsp;&nbsp;&nbsp; <input type=text name='start'>
          <br>&nbsp;&nbsp;&nbsp; <input type="input" name='end'>
          <br> After filling the form please press <br>
          <p><input type='submit' value="submit"></p>
      </form>
    </body>
</html>
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*

* 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

<SearchForm>

<fields>
  <field name="dc:creator" type="string">
    <rel-ops selected="equal">
      <rel-op name="equal" symbol="" position="infix"/>
      ...
    </rel-ops>
    ...
  </field>
</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" appearance="min">
  <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:repeat>
XForms Example – SCHOLNET Project

```xml
<SearchForm>
  ...

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

</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" appearance="min">
  <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:repeat>
```
XForms Example – SCHOLNET Project

```xml
<SearchForm>
  ...
</SearchForm>

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

<SearchForm>
  ...
</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:group nodeset="rel-ops" appearance="min">
  <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:group>
</xforms:repeat>
```
How to use XForms?

Edit XForms Model
(Form Dialog Manager)

Design time

Run time

Mapping of XForms Model
⇔
Data Model
(VizCo *)

Transform XForm Model
⇔
Client Interface

formsPlayer © x-port


© Fraunhofer IPSI
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)

Intelligent User Interfaces
Semantic Web Application Models (2)

Semantic Domain Model

User Interface Layer

Conceptual UI Model

Application Layer

Web User Interface

Agent Interface

Software Agent

Web Client (HTML)

Ontology

What ✓ How ✓
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 models**
  - 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
Example
“Different types of travel can be defined starting from the concept Travel defined in another global schemes”

```xml
<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>
```

“definition of new subclasses”

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)

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)

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”

```
<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>
```

“definition of new superclass”
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

Diagram:
- User Interface Layer
- Application Layer
- Web User Interface
- Agent Interface
- Conceptual UI Model
- Semantic Domain Model
- Software Agent
Operationalization: Model Authoring + Mapping Definition → FairsNet

System Tools Suite

**Publishing Tools**
- Form Dialog Manager
- Menu Manager

**Task-specific Tools**
- Booking Manager
- User Manager

Web User Interface
Agent Interface
Conceptual UI Model
Semantic Domain Model
Web Services

Software Agent
Process Model Manager

© Fraunhofer IPSI
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!