A Framework to Specify Hypertext Interfaces for Ontology Engineering

FALQUET, Gilles, MOTTAZ JIANG, Claire-Lise
A Framework to Specify Hypertext Interfaces for Ontology Engineering

Gilles Falquet & Claire-Lise Mottaz
C.U.I. University of Geneva
Context

Creation and maintenance of ontologies in an organizational memory

Proposal

- Use the hypertext paradigm to access the ontology
- Generate specific hypertext interfaces
  - domain/task/user dependent
- A language to specify hypertext interfaces
  - based on hypertext views on databases
  - executable specifications (immediate) implementation
Outline

- Ontologies and organizational memories
- Ontology engineering tools and their interfaces
- Hypertext views on databases
- Specifying ontology engineering interfaces
Designing Ontologies

♦ Formalization of a domain
  ♦ modelling problem
  ♦ refer to existing documents, data

♦ Consensualization
  ♦ obtain a consensual view of the domain
  ♦ solve conflicts, reconcile different views

♦ Apply specific methodologies
  ♦ IBIS (issues, positions, arguments, endorsements)
  ♦ Voting, etc.
Extended Ontology

Ontology engineering requires

 présente

- Ontology storage structures

- Methodology/engineering structures
  - discussion, argumentation, etc.
  - references to documents/data

- Reference ontologies
  - common concepts, general knowledge
Organizational Memory Structure

- Extended Ontology
  - Domain ontology (formal knowledge)
  - Auxiliary ontologies
  - Methodology/design process support

- Documents
  - non-formalized knowledge, facts

- Data
  - formalized factual information
Existing tools and their interfaces

- Ontology editors (single user)
  - OntoWeb, Protégé, OilEd

- Collaborative tools
  - WebODE, OntoLingua, CO4
  - Discussion mechanisms, voting

Interfaces

- Web interfaces (form-based)
- in terms of the formal language
- sometimes: analysis tools (WebGrid, InfoLens)
Ontology visualization

- Using existing visualization techniques
  - OntoBroker => hyperbolic trees

- Specific techniques
  - SemNet => 3D graph showing semantic distances
Protégé - OWL plugin
Rice: hierarchy + graph
Problems with existing tools

- Work only on the formalized definitions
  - exceptions: WebODE, Terminae

- Store the result of consensualization
  - do not support the conflict resolution process

- Not connected to the information sources

- Provide only one interface
  - A single interface cannot fit all domains / applications
  - Customization requires programming (plugins, Java Beans)
Using Database Interfacing Technologies?

- Store the ontology in a database
  - database schema = meta ontology
  - data = concept definitions

- Use existing database interfacing tools
  - table display, query form generators, etc.

Does not work well because ...

- DB tools are intended for complex schemas and simple data
- Ontological data are complex (graph, trees, syntax trees, etc.)
Hypertext and Knowledge Management

- Historically related (NLS, KMS, etc.)
- HT systems that embed KR (MacWeb, etc.)
- Graph/associative nature
Hypertext views on databases

- Principles
- Represent the content of a DB as a hypertext
- Navigation replaces querying

```
tables, rows, columns
```

```
nodes (pages), hyperlinks
```
Declarative approach

- Hypertext nodes are instances of node schemas

node schemas

specify

tables, rows, columns

node instances
## Nodes schema

<table>
<thead>
<tr>
<th>name [parameters]</th>
</tr>
</thead>
<tbody>
<tr>
<td>contents</td>
</tr>
<tr>
<td>+</td>
</tr>
<tr>
<td>links</td>
</tr>
</tbody>
</table>

| data selection    |
A node schema

<table>
<thead>
<tr>
<th>Artists_after[date]</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ &lt;artist&gt;</td>
</tr>
</tbody>
</table>
|     <name>name </name>
|     <birth>birthdate</birth>
|     </artist>     |
| }                  |
| artists : birthdate >= date |

Works_by [artist_ID]
Two instances

Artists_after[1860]

Charles Sheeler (born 1883) [works]
Georges-Pierre Seurat (born 1859) [works]
Kasimir Malevich (born 1878) [works]
Kurt Schwitters (born 1887) [works]
Pablo Picasso (born 1881) [works]
Pierre Bonnard (born 1867) [works]
Piet Mondrian (born 1872) [works]
Robert Delaunay (born 1885) [works]
Vincent van Gogh (born 1853) [works]

Artists_after[1880]

Charles Sheeler (born 1883) [works]
Kurt Schwitters (born 1887) [works]
Pablo Picasso (born 1881) [works]
Robert Delaunay (born 1885) [works]
Another node schema

<table>
<thead>
<tr>
<th>Works_by[artist]</th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
</tr>
<tr>
<td>&lt;image&gt;picture&lt;/image&gt;</td>
</tr>
<tr>
<td>&lt;desc&gt;</td>
</tr>
<tr>
<td>title, c_date, support,</td>
</tr>
<tr>
<td>height, width</td>
</tr>
<tr>
<td>&lt;/desc&gt;</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>work : author = artist</td>
</tr>
</tbody>
</table>
Works_by[1]

Charles Sheeler (born 1883) [works]
Georges-Pierre Seurat (born 1859) [works]
Kasimir Malevich (born 1878) [works]
Kurt Schwitters (born 1897) [works]
Pablo Picasso (born 1881) [works]
Pierre Bonnard (born 1867) [works]
Piet Mondrian (born 1872) [works]
Robert Delaunay (born 1885) [works]
Vincent van Gogh (born 1853) [works]

The Breakfast Room c. 1930-31
Oil on canvas 62 7/8 inches x 44 7/8 inches

La partie de croquet 1892
Oil on canvas 130cm x 162cm
Hypertext model

- Content model: XML or HTML elements
- Link model: (more than the Web model)
  - reference links (href)
  - inclusion links (transclusion)
    - to build complex nodes
- expand-in-place links
  - dynamic inclusion
Hypertext views on knowledge bases

- Store the extended ontology in a DB
- Store the documents and data in a DB
- Represent the ontology language in a DB schema
- Define hypertext views on this DB
Exemple: viewing a DL ontology

- Database schema of the OM

  table Concept (id, defCategory, ...)
  table SubConcept (super, sub)
  table RoleRestriction(id, appliesTo, role, range, ...)

Example: subconcepts and roles

```
subConceptsOf[c]
  c
    { sub }
SubConcept: super = c
rolesOf[c]
  c
    { quantifier, role, range, ... }
RoleRestriction: appliesTo = c
```

*subConceptsOf*[c]  
  c
    { sub }
SubConcept: super = c

*rolesOf*[c]  
  c
    { quantifier, role, range, ... }
RoleRestriction: appliesTo = c

href rolesOf[sub]
href subConceptsOf[c]
An Instance: subConceptsOf[POTABLE-LIQUID]
An Instance: roleRestrictionsOf[WINE]
With expand-in-place links

```
node subConceptsOf[c]
    { sub, expand href subConceptsOf[ sub ]([open]) } from SubConcept selected by super = c
```
subConceptsOf[POTABLE-LIQUID]

<table>
<thead>
<tr>
<th>Concept</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>WINE</td>
<td>[open]</td>
</tr>
<tr>
<td>WATER</td>
<td>[open]</td>
</tr>
<tr>
<td>TEA</td>
<td>[open]</td>
</tr>
<tr>
<td>COFFEE</td>
<td>[open]</td>
</tr>
<tr>
<td>FIZZY-DRINK</td>
<td>[open]</td>
</tr>
</tbody>
</table>
subConceptsOf[POTABLE-LIQUID]
subConceptsOf[POTABLE LIQUID]
subConceptsOf[POTABLE-LIQUID]
With inclusion links

\[
\begin{array}{|c|}
\hline
\text{subConceptsOf}[c] \\
\hline
\text{c} \\
\quad \{ \text{sub exp} \} \\
\hline
\text{SubConcept} : \\
\quad \text{super} = \text{c} \\
\hline
\end{array}
\]

include

\[
\begin{array}{|c|}
\hline
\text{subConceptsOf}[\text{sub}] \\
\hline
\end{array}
\]
subConceptsOf[POTABLE-LIQUID]

<table>
<thead>
<tr>
<th>WINE</th>
<th>LATE-HARVEST</th>
<th>EARLY-HARVEST</th>
<th>DESSERT-WINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>WATER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEA</td>
<td>GREEN-TEA</td>
<td>OOLONG-TEA</td>
<td>BLACK-TEA</td>
</tr>
<tr>
<td></td>
<td>DARJEELING-SPRING</td>
<td>DARJEELING-SUMMER</td>
<td>ASSAM</td>
</tr>
<tr>
<td></td>
<td>CEYLON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COFFE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIZZY-DRINK</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

subConceptsOf[WINE]

subConceptsOf[TEA]

subConceptsOf[BLACK-TEA]
Building Complex Nodes

- Inclusion links = mechanism to build complex heterogeneous nodes

```
viewConcept
  ───> superConceptOf
         ├── inc
         └── inc

subConceptOf
  ───> rolesOf
         ├── inc
         └── inc

rolesOf
  ───> atomicConcept
          ───> Description
                    ───> comments

comments
  ───> inc
          ───> comments
```

Specification Schema

- Graphical representation of the hypertext view specification
- Evaluating properties of the interface design
Interface Design Methodology for OM

- No "one size fits all" interface
- Design must be adapted to each situation

- Generic interface design process
  1. hypertext view on the ontology
  2. view on the extended ontology
  3. view on the whole OM (documents, data)
1. Ontology Interface Design

Tasks: authoring, exploring, checking

1. Initial structure
   ◆ basic navigation: sub-concept hierarchy, roles, arguments (unions, intersections)

2. Extended navigation patterns
   ◆ proximity, affinity, syntactic distance

3. Assembling elements
   ◆ building complex nodes

4. Update nodes
Example: A Complex Node Instance
2. Views on the extended ontology

- Accessing the methodology support data

  - Annotations

  - Discussion, argumentation

  - etc.
Extended ontology

Term

Concept
  +id
  subConceptOf

Intersection
  +arguments

RoleRestriction
  +min, max
  +quantifier
  +role
  +range

AtomicConcept

Argument

PointOfView

Note
Example: managing notes

- Nodes on the extended ontology

**node** NotesOn[c]

{ show the annotation text and author }

**from** Annotation selected by onConcept = c

**node** WriteNoteOn[c]

**active href** ViewConcept[c] (  
**set** onConcept = c, **set** text = *textfield*(),  
**set** author = [USER]  
**on** "save" **do** insert into Annotation)
A Node with Note Management

Subconcepts of POTABLE-LIQUID

From: Jacques
The role restriction on MAKER should be removed ...
delete

From: Giles
This definition should include a ...
delete

[new note]
WATER [roles] [notes] [new note]

TEA [roles] [notes]
Add a subconcept 'red tea' to ....

COFFE [roles] [notes] [new note]
3. Views on the whole OM

- Accessing related data
  - nodes on other database tables

- Accessing related documents
  - stored in databases
  - stored elsewhere (store references into a database)

- Crossing the abstraction gaps
  - e.g. interconnecting data and concepts
Example: a Knowledge Based eBook

- Simple ontology model
- Information fragments (texts, images)
- Semantic indexing of fragments
  - O-F links: example-of, requires, exercise on, ...
- Reading and Writing interface
- Re-construction of complex documents (link inference)
Automate à états fini

Un AF est composé de

1. un alphabet de symboles reconnus $A = \{s_1, s_2, \ldots, s_n\}$
2. un ensemble d'états $Q = \{q_1, q_2, \ldots, q_m\}$ (les ronds)
3. un état initial $q_1$ de $Q$
4. un ensemble $F$ d'états finals inclus dans $Q$
5. une fonction de transition $d$ de $Q \times A$ dans $Q$ (les flèches)

Concepts liés

Machine de Turing

Acceptation d'une chaîne par un AF

Tous les liens

consomme --> chaîne [Open]
exemple --> AEF-ex1 [Open]
estension --> mdt [Open]
référence --> accept-chaîne-a [Open]

[Nouveau Lien]

Liste des fragments
Conclusions

- Sophisticated user interfaces for collaborative ontology engineering can be specified by defining simple hypertext node schemas.
- Schema definition language is concise and simple, the interface can easily be extended and maintained by non programmers and non database specialists.
Conclusions (cont.)

- This declarative language is also well adapted to an incremental approach of interface design. Hence, we proposed an interface design methodology

- Integration capability offered by hypertext views: ontology - data - documents

- Implementations: with the Lazy hypertext view system on top of relational databases

- Future implementations: on top of RDF databases