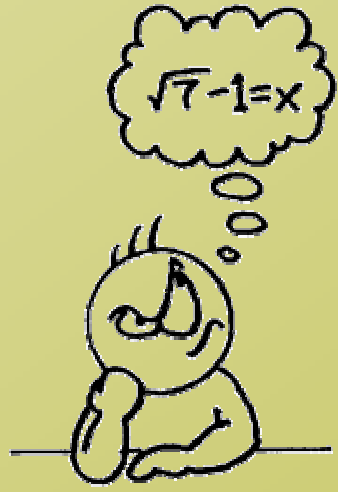
UNIVERSITÄT LEIPZIG

„Agile Knowledge Engineering – Methodology, Concepts and Applications“

Sören Auer
Uni Leipzig GK „Wissensrepräsentation“ &
Abt. Betriebliche Informationssysteme
auer@informatik.uni-leipzig.de

Outlook



Problem: Enable users, domain experts and knowledge engineers to collaborate efficiently in building knowledge bases?

Possible solution: **Agile Knowledge Engineering**

Agile methodologies gain increasing importance:

- Software Engineering - Extreme Programming
- Web Content Management – Wiki Wiki Webs

For Semantic Web data **an agile KE methodology amplifies the need to keep track of changes** and to rapidly query, aggregate and visualize information.

UNIVERSITÄT LEIPZIG

Agenda

1. **RapidOWL: Agile knowledge engineering methodology**
2. Enabling agility: Ontology evolution and data migration
3. RDF query caching: Accelerate querying and aggregation
4. Powl: framework for Semantic Web application development
5. OntoWiki: Semantic Web application supporting RapidOWL
6. “Vernetzte Kirche“: RapidOWL use case

Existing Methodologies

Uschold's methodology

- Identification of the ontology's purpose
- Capture key concepts & relationships in unambiguous textual form
- Mapping onto precise terminology
- Coding in formal KR language, integrating with existing ontologies
- Evaluation and documentation for modification and reuse

Grüninger and Fox

- Similar to Uschold
- Check competency of the ontology (informal questions)

Methontology

- More complex (incl. project management and support activities)
- Waterfall based development process model
- Iterates through phases: specification, knowledge acquisition, conceptualization, integration, implementation, evaluation, documentation

Problems of Existing Methodologies

Task is starting point (i.e. construct ontology with usage scenario in mind):

- Requires significant initial effort
- Makes changes to and reuse of the resulting ontologies inherently hard (cf. Holger Knublauch – XP.K)

Analogous to SW Development Methodologies:

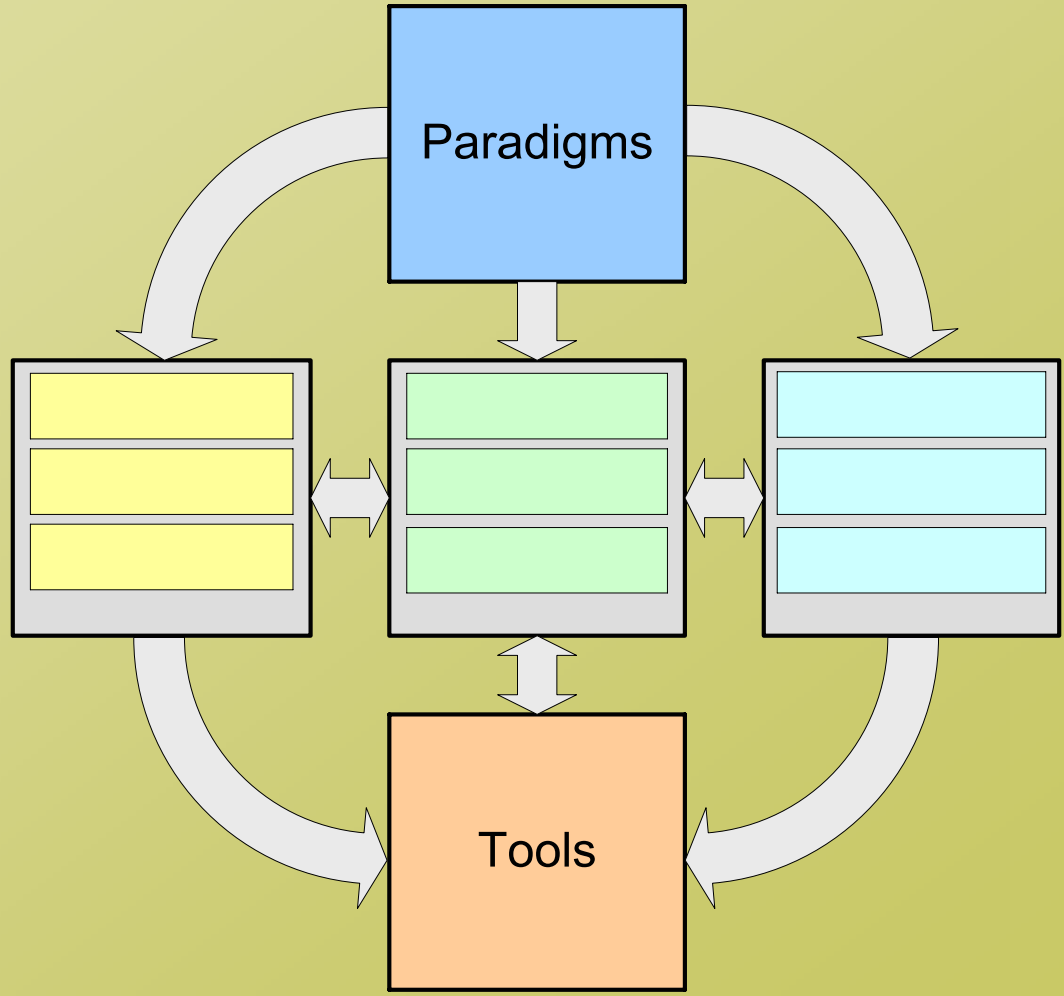
- Do not respect specific KE characteristics

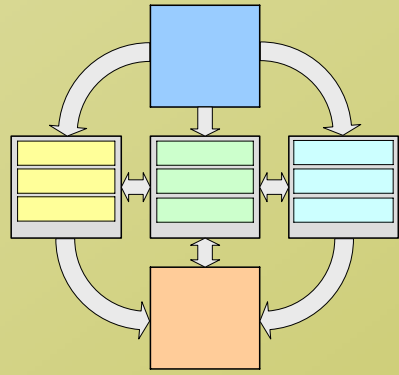
→ **virtually not used in practice** (cf. Report of the EMISA-Forum)

Specific Characteristics of KE

- Not a Business in itself
- No Unique Knowledge Serialization
- Spatial Separation of Actors
- Involvement of a Large Number of Actors
- Single, precise usage scenario either not (yet) known or not (easily) definable

RapidOWL - An Agile Methodology





RapidOWL

Sketch of an agile KE methodology inspired by XP:

Paradigms:

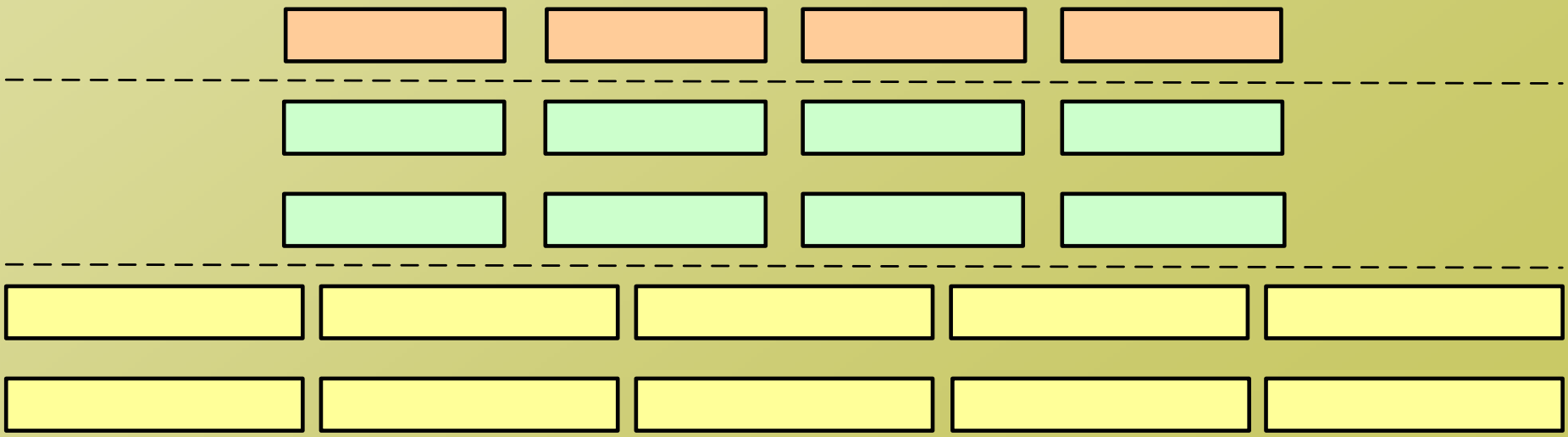
- Generic architecture of knowledge-based systems
- RDF Data Model
- Web Technologies

Tools:

- Powl, Protégé, OntoWiki

Models:

- Semantic Web Layer
  (RDF, RDF-Schema, OWL, upper-level & domain ontologies)



RapidOWL Collaboration

Developer
Generates views
Manages releases
Maintains the system

Domain expert
Modells and refines ontology
Integrates existing information

Transparent RapidOWL Knowledge Base
(RDF-S/OWL ontology and instance data)

Domain expert & User
Contributes instance data
Provides feedback
Votes about correctness

Knowledge Engineer
Supervises modelling and evolution
ensures modelling standards

UNIVERSITÄT LEIPZIG

Agenda

1. RapidOWL: Agile knowledge engineering methodology
2. **Enabling agility: Ontology evolution and data migration**
3. RDF query caching: Accelerate querying and aggregation
4. Powl: framework for Semantic Web application development
5. OntoWiki: Semantic Web application supporting RapidOWL
6. “Vernetzte Kirche“: RapidOWL use case

RDF Basics

Literal: string with either language identifier (plain) or datatype (typed).

Blank Node: Set of blank nodes, set of URI references, and set of literals are pairwise disjoint. Otherwise, set of blank nodes is arbitrary.

A **Statement** is a triple *(S,P,O)*, where

- *S* is either URI reference or blank node (Subject).
- *P* is URI reference (Predicate).
- *O* is either URI reference or literal or blank node (Object).

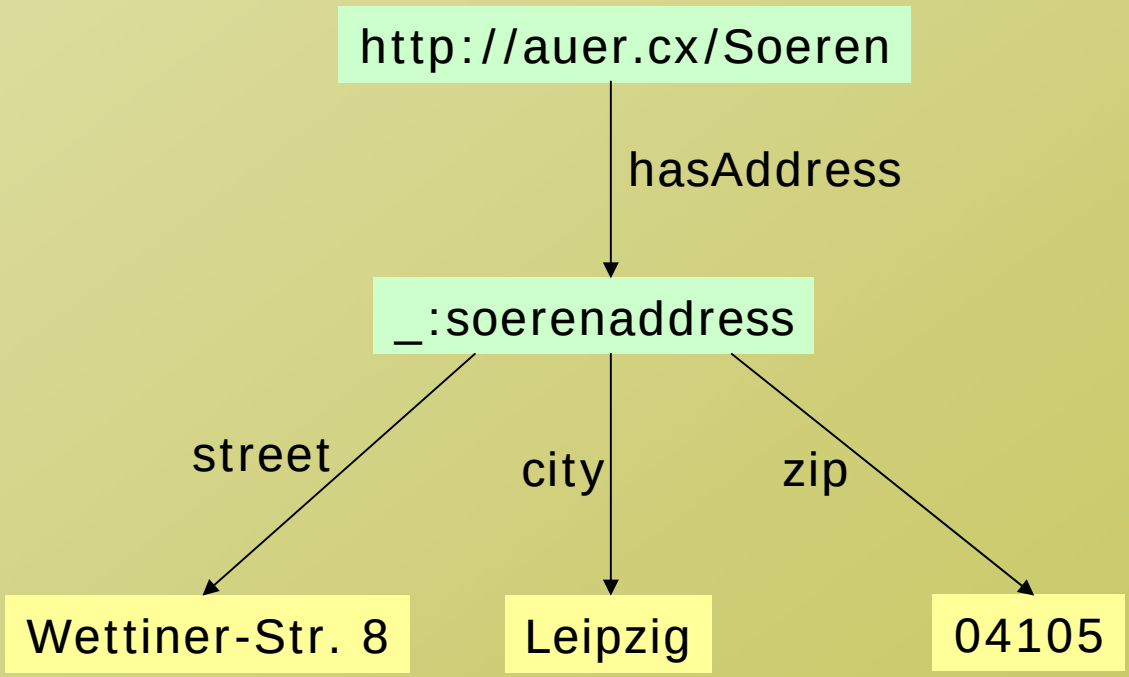
A **Graph** is a set of statements.
set of nodes of a graph = set of subjects and objects
blank nodes of a graph = subset consisting only of blank nodes.

Graphs *G* and *G'* are **equivalent** ↔ bijection *M* between the sets of nodes:

1. *M* maps blank nodes to blank nodes.
2. *M(lit)=lit* for all literals *lit* which are nodes of *G*.
3. *M(uri)=uri* for all URI references *uri* which are nodes of *G*.
4. Triple *(s,p,o)* is in *G* ↔ triple *(M(s),p,M(o))* is in G'.

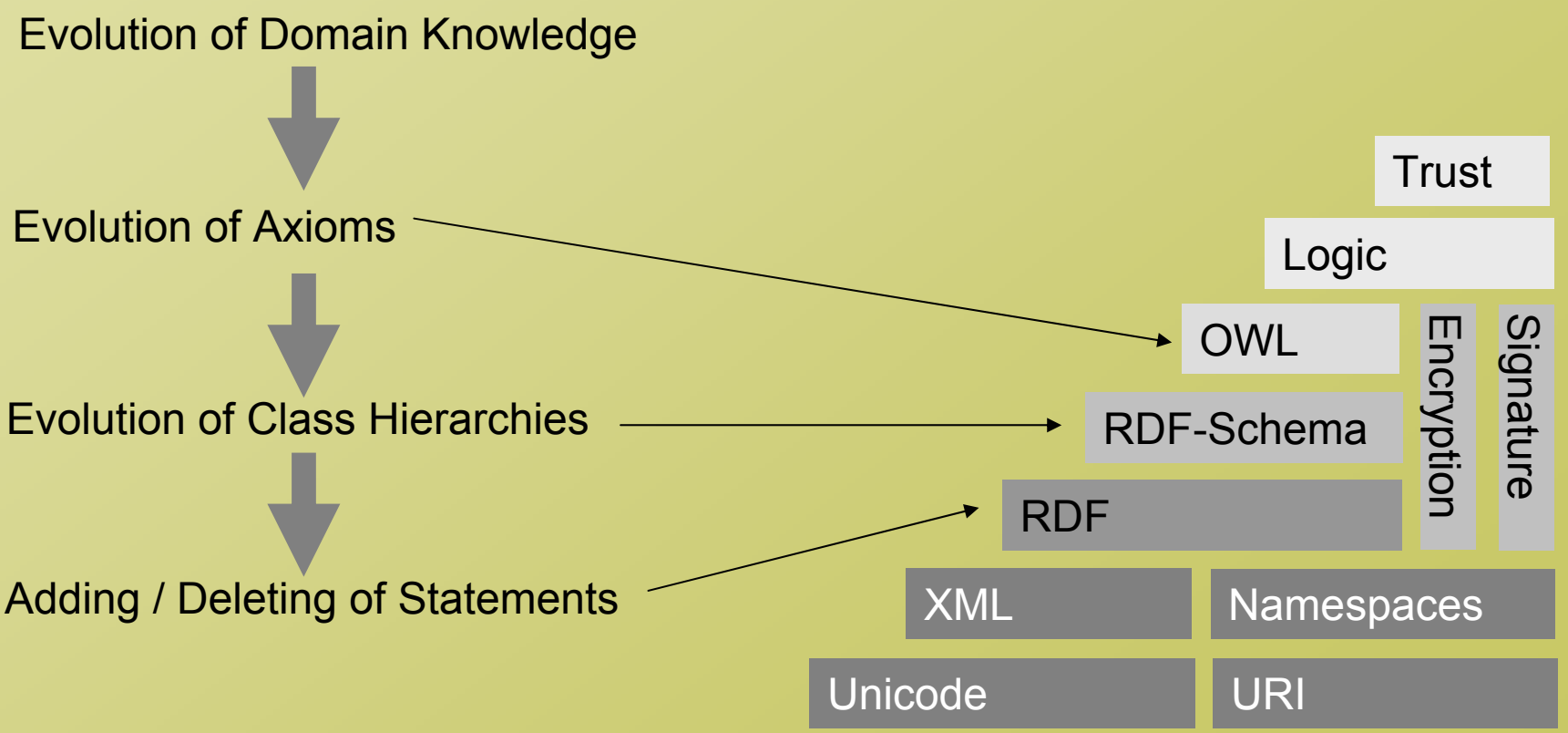
Example

Subject	*Predicate*	*Object*
http://auer.cx/Soeren	hasAddress	_:soerenaddress
_:soerenaddress	street	„Wettiner-Str. 8"
_:soerenaddress	city	„Leipzig"
_:soerenaddress	zip	„04105"



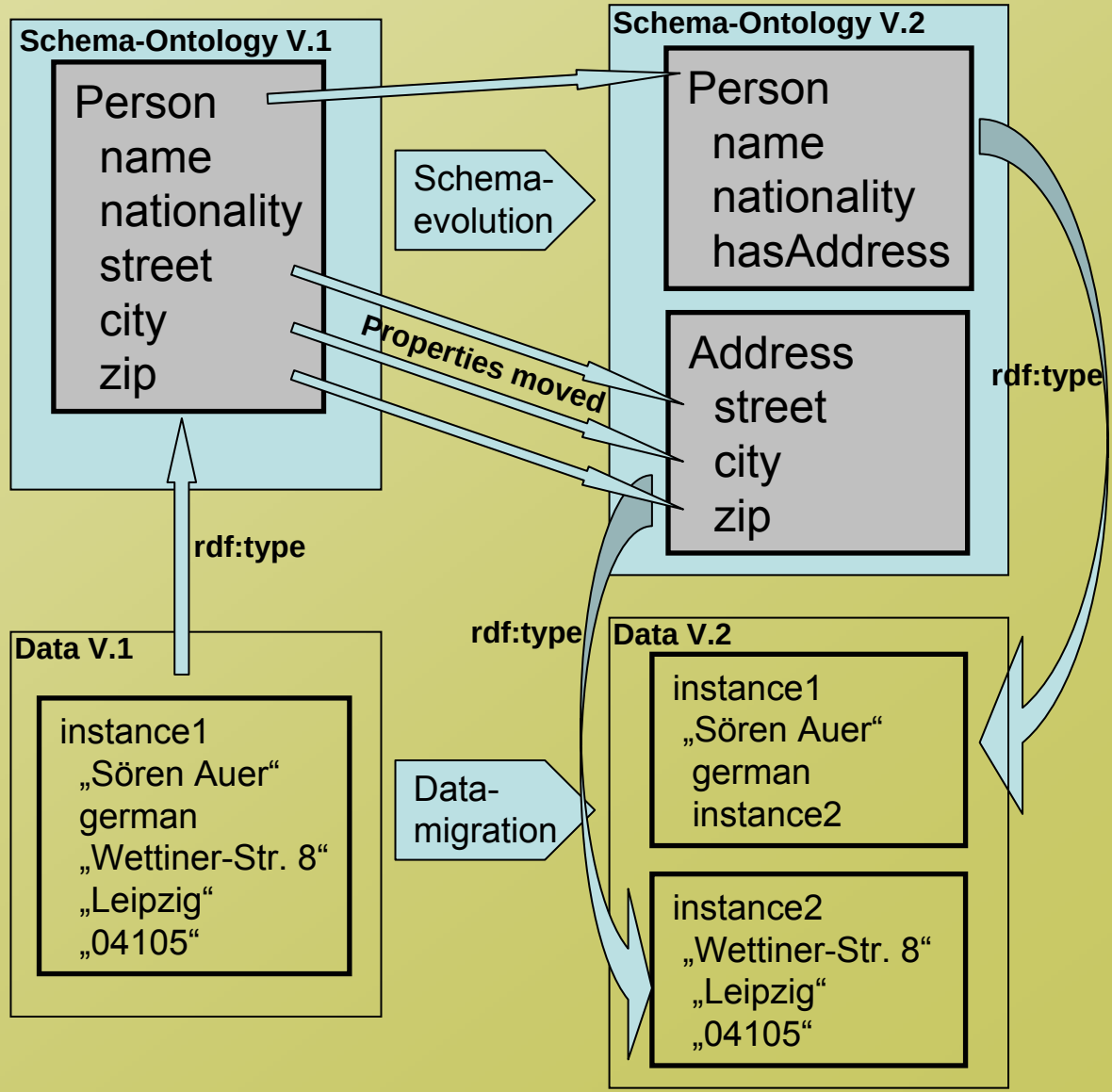
Problem 1: Consistent Evolution

Consistent Evolution on different levels of Knowledge Representation



UNIVERSITÄT LEIPZIG

Problem 2: Distributed Evolution



Respect the distributed character of the Web

```
Person rdf:type    owl:Class
name   rdfs:domain Person
nation rdfs:domain Person
street rdfs:domain Person
city   rdfs:domain Person
zip    rdfs:domain Person
```

Atomic Changes

A graph is **atomic** if it may not be split into two nonempty graphs whose blank nodes are disjoint.

A atomic graph C_G is said to be an **Positive Atomic Change** on a graph G if the sets of blank nodes occurring in statements of G and C_G are disjoint.

Example: **`Adress rdf:type owl:Class`**

$\boldsymbol{Apl^{+}(G, C_G)} = G \cup C_G = G'$ (C_G is applied to G with result G')

A subgraph C_G of G is said to be a **Negative Atomic Change** on a graph G if C_G is atomic and contains all statements of G whose blank nodes occur in the statements of C_G.

Example: **`street rdfs:domain Person`**

$\boldsymbol{Apl^{-}(G, C_G)} = G \setminus C_G = G'$ (C_G is applied to G with result G')

General Changes

Positive and Negative Atomic Changes = **Changes of Level 0**

Changes of higher levels are defined inductively:

At = set of atomic changes.

General changes are sequences over *At:*

Changes(At) = smallest set containing the empty sequence *()* and closed with respect to the following condition:
if $\{C_1 , \dots C_k\} \subseteq$ Changes(At) $\cup$ At, then $(C_1 , \dots , C_k) \in$ Changes(At).

Changes of level at least n - Ch(n) - are defined inductively:

Every change has a level at least 0, i.e. Ch(0) = Changes(At).

If $C_1 , \dots , C_k$ are changes in *Ch(n)*, then $(C_1, \dots , C_k) \in Ch(n+1)$.

A **change C is of level (exactly) n** if $C \in Ch(n) \setminus Ch(n+1)$.

Application functions App^+ , App^- may be extended to *App(G,C)*
first argument is a graph, second argument a change.

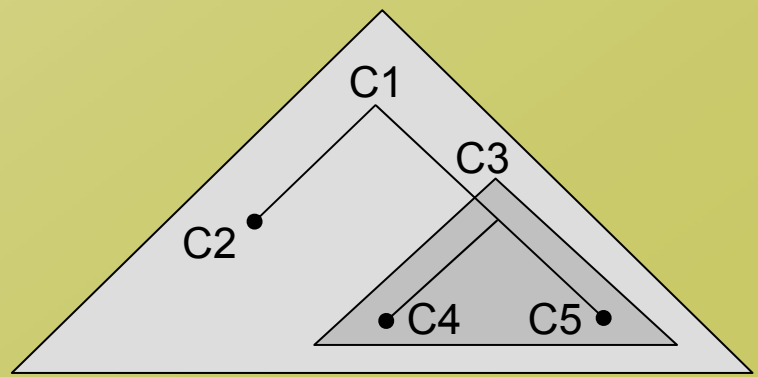
App is recursively defined on the level of the second argument *C*.

Example: General Change

```
Resource changed                                    (C1)
  Resource classified                               (C2)
    http://auer.cx/Soeren  nationality  German
  Labels changed                                    (C3)
    Label removed                                   (C4)
      http://auer.cx/Soeren  rdfs:label  "Soeren
  Auer"
    Label added                                     (C5)
      http://auer.cx/Soeren  rdfs:label  "Sören Auer„
```

Corresponding sequence of atomic changes:

(C2,C4,C5)



Statement Level Compatibility

Tracking changes enables linear undo / redo.
To selectively rollback changes - compatibility concept needed:

Compatibility of a Positive Atomic Change with a Graph:
A Positive Atomic Change C_G is compatible with a graph G', iff C_G is not equivalent to some subgraph of G'.

Compatibility of a Negative Atomic Change with a Graph:
A Negative Atomic Change C_G is compatible with a graph G', iff C_G is equivalent to some subgraph of G'.

Compatibility of a Compound Change with a Graph:
A compound change $C_{G'}$ is compatible with a graph G, iff

- the first atomic change in the corresponding sequence of atomic changes $(C_1,...,C_n)$ is compatible with G and results in G^1
- every following atomic change C_i $(1<i<=n)$ from the sequence is compatible with the intermediate graph G^{i-1} and its application results in G^i.

Representation and Annotation

Changes can be represented and annotated using RDF itself:

```
C1  rdf:type           log:Change
C1  log:Action         log:ResourceChanged
C1  log:User           http://auer.cx/Soeren
C1  log:DateTime       "20050320T16:32:11"
C1  log:Documentation  "Nationality added and name typing fixed"

C2  rdf:type           log:Change
C2  log:Action         log:Classified
C2  log:DateTime       "20050320T16:32:11"
C2  log:Documentation  "Soeren Auer is of German Nationality!"
C2  log:ParentAction   C1
C2  log:added          S1
…

S1  rdf:type           rdf:Statement
S1  rdf:subject        http://auer.cx/Soeren
S1  rdf:predicate      hasNationality
S1  rdf:object         German
…
```

What have we got so far:

- Description of arbitrary changes of different levels of detail
- Notion of compatibility on the level of statements

What is missing:

- Only considered arbitrary RDF knowledge bases, what about higher conceptual levels, ontologies?
- Can we classify changes on higher concetual levels?

Basic Evolution Patterns for Classes

class added (-)
class removed (+)
class equivalence change
…
restriction change

- cardinality change
  - lowerbound change
    - lowerbound added (+)
    - lowerbound removed (-)
    - lowerbound modified
      - lowerbound lower (-)
      - lowerbound higher (+)

…
superclass change

- superclass added (+ - inheritance of restrictions)
- superclass removed (+ usage of inherited properties)
- superclass modified
  - superclass changed to subclass (+)
  - superclass changed to superclass (+)

From WonderWeb Deliverable D14
Insufficient to capture
the intentions of changes!!!

Example: Class Deletions

Class deletion effects:

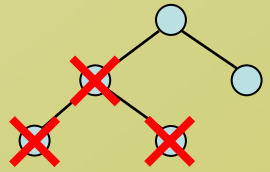
- former instances less specifically typed,
- former direct subclasses become independent top level classes,
- properties having the class as domain become universally applicable,
- properties having the class as range will lose this restriction,

How to deal with:

- Instances
  - Reclassify to superclass
  - Delete
- Subclasses
  - Delete
  - Reassign in the class hierarchy
- Properties having the class as domain or range
  - Extend domain and range
  - Restrict domain and range

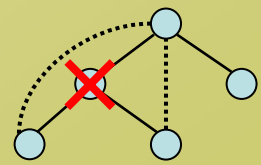
Ontology Evolution Patterns

Delete complete subtree:



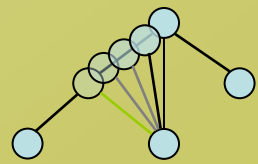
- delete instances
- delete subclasses
- delete directly attached properies

Cut class off



- delete instances
- reassign subclasses to superclass
- change domain of directly attached properies to superclass

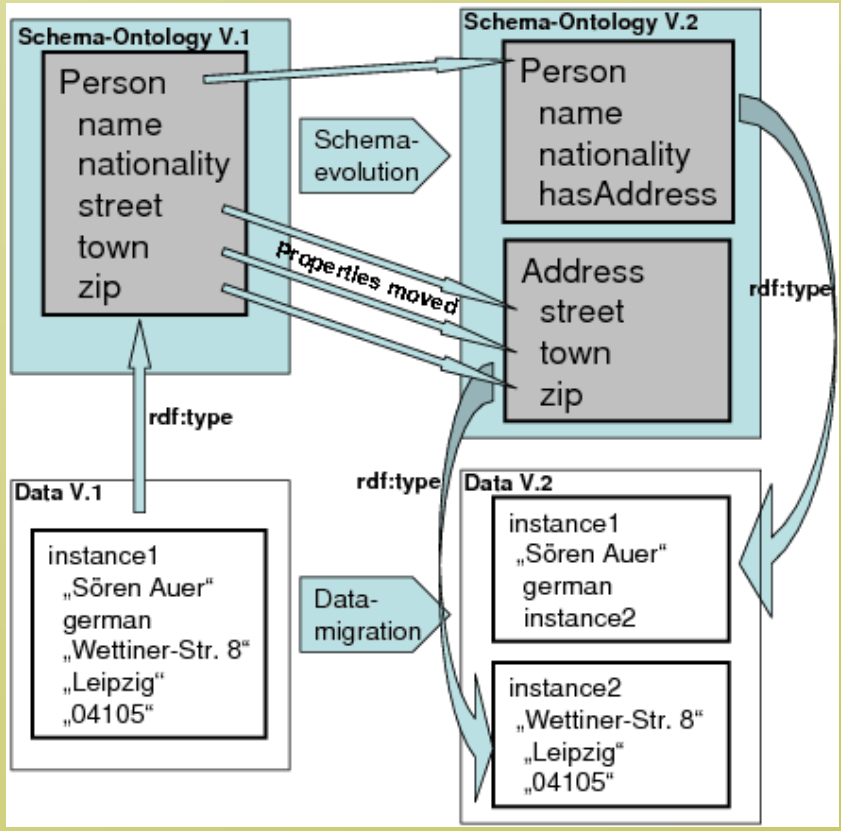
Merge class with superclass



- reclassify instances to superclass
- reassign subclasses to superclass
- change domain of directly attached properies to superclass

UNIVERSITÄT LEIPZIG

Data Migration



Evolution Pattern: Move Property

```
foreach triple (?i1,rdf:type,C1) in G
  find triple (i1,P1,?v) in G
  foreach triple (i1,P2,?i2) in G
    add triple (i2,P1,v) to G
    del triple (i1,P1,v) from G
```

Evolution Pattern: Slit Class

```
add triple (C1,rdf:type,owl:class) to G
foreach triple (?i1,rdf:type,C) in G
  create new instance identifier i
  add triple (i,rdf:type,C1) to G
  add triple (i1,P,i) to G
foreach moved property P
  PropertyMove(C,C1,P)
```

UNIVERSITÄT LEIPZIG

Agenda

1. RapidOWL: Agile knowledge engineering methodology
2. Enabling agility: Ontology evolution and data migration
3. **RDF query caching: Accelerate querying and aggregation**
4. Powl: framework for Semantic Web application development
5. OntoWiki: Semantic Web application supporting RapidOWL
6. “Vernetzte Kirche“: RapidOWL use case

Problem 2: Rapid RDF Querying

Crucial to rapidly **extract, aggregate and rearrange RDF data** for visualization, editing, syndication

- usually time consuming and resource demanding (real life data, scale well)

Compared to RDBMS, **triple stores are slower**:

- fields in relational database schema are typed (indexed efficiently)
- in triple stores, this efficiency is lost to flexibility

In a short time period usually only small KB changes

=> high probability that a query to the KB will return the same result

Queries are executed often

Query analysis shows, on which KB changes different results will be returned.

Meanwhile **caching:**

=> improved usability and response times of SW applications

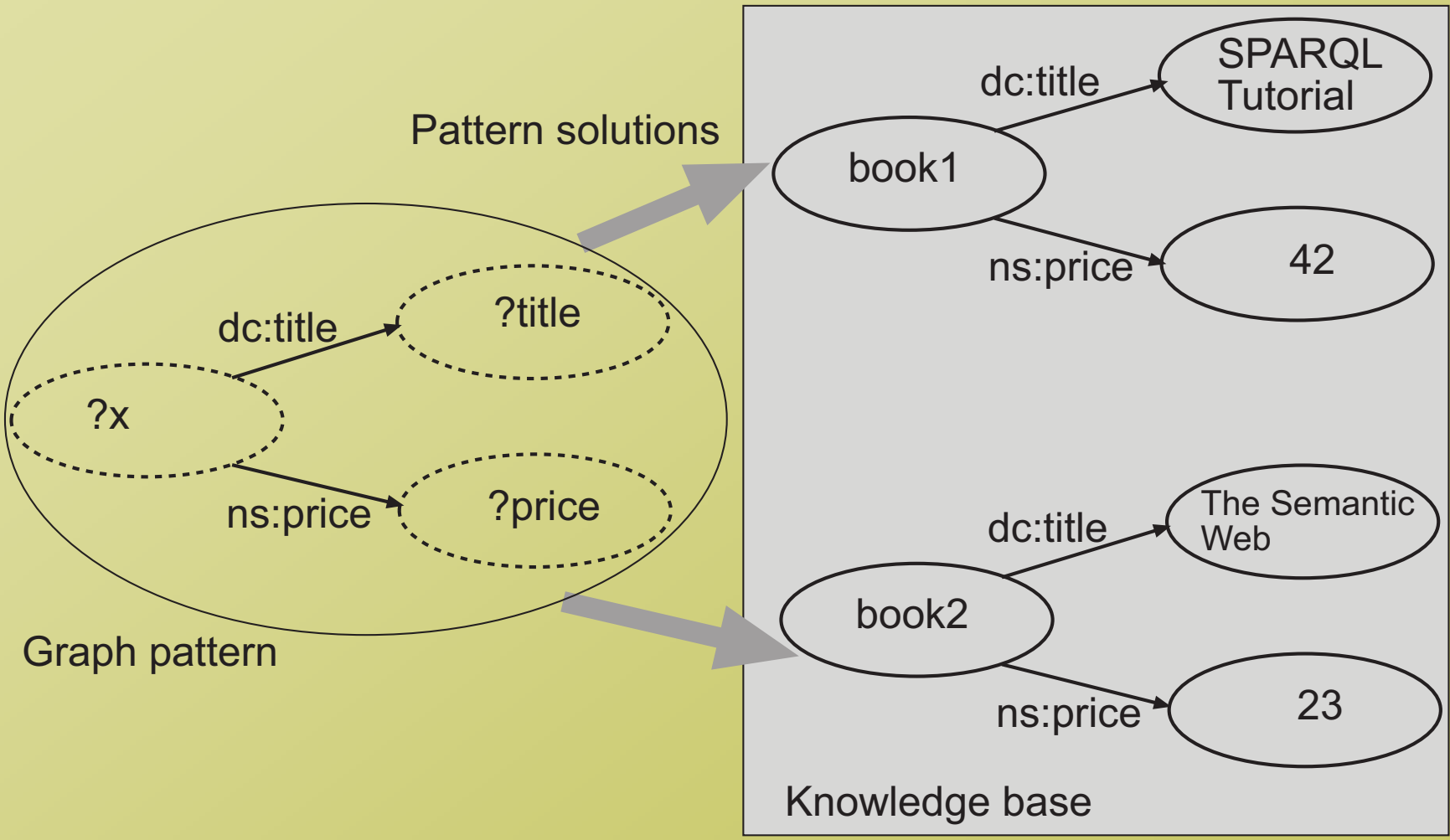
Example

```
book1  dc:title  "SPARQL Tutorial"
book1  ns:price  42
book2  dc:title  "The Semantic Web"
book2  ns:price  23
```

SPARQL query to return the title and the price of resources, whose price is less than 30:

```
SELECT ?title ?price
  WHERE (?x dc:title ?title)
        (?x ns:price ?price)
  AND ?price < 30
```

Graph Patterns and Solutions



Proposition: Solution Invariance

G: graph

C: atomic change on G leading to G'.

P: graph pattern

- every contained triple pattern contains at least one URI reference
- sets of URI references occurring in the atomic changes of C and P are disjoint.

Then a substitution S is a pattern solution of P on G iff S is a pattern solution of P on G'.

Benchmark 100k Queries

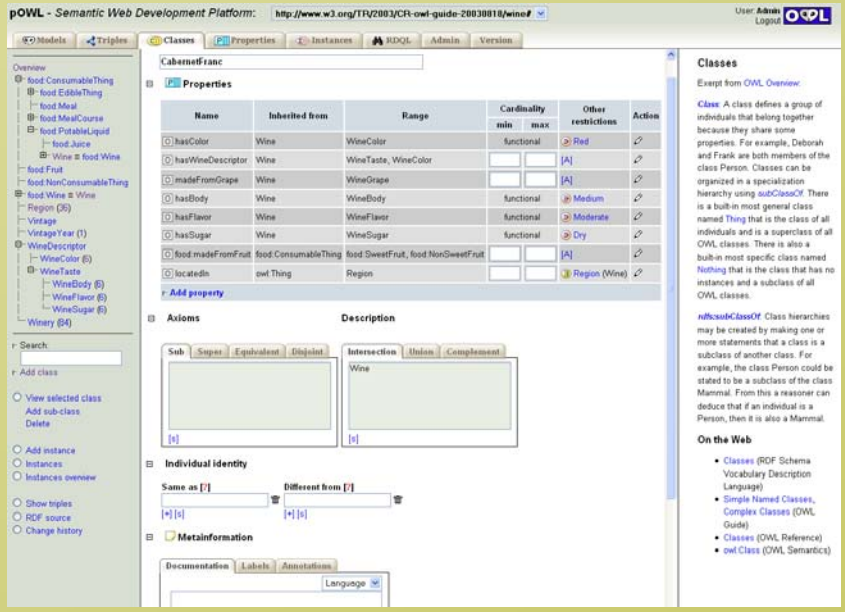
Update/query ratio	Time without caching	Time with caching	Perf. gain
0.01	1283s	412s	68%
0.001	1247s	387s	69%
0.0001	1191s	322s	73%

Agenda

1. RapidOWL: Agile knowledge engineering methodology
2. Enabling agility: Ontology evolution and data migration
3. RDF query caching: Accelerate querying and aggregation
4. **Powl: framework for Semantic Web application development**
5. OntoWiki: Semantic Web application supporting RapidOWL
6. “Vernetzte Kirche“: RapidOWL use case

Tool support: Powl

- **Opensource**,
- **Web-based**, for most deployed Web technologies (PHP/MySQL)
- Semantic Web **development platform**
- Allows parsing, storing, querying, manipulating, versioning, serving and serializing RDF-S and OWL
- In a **collaborative** web enabled environment



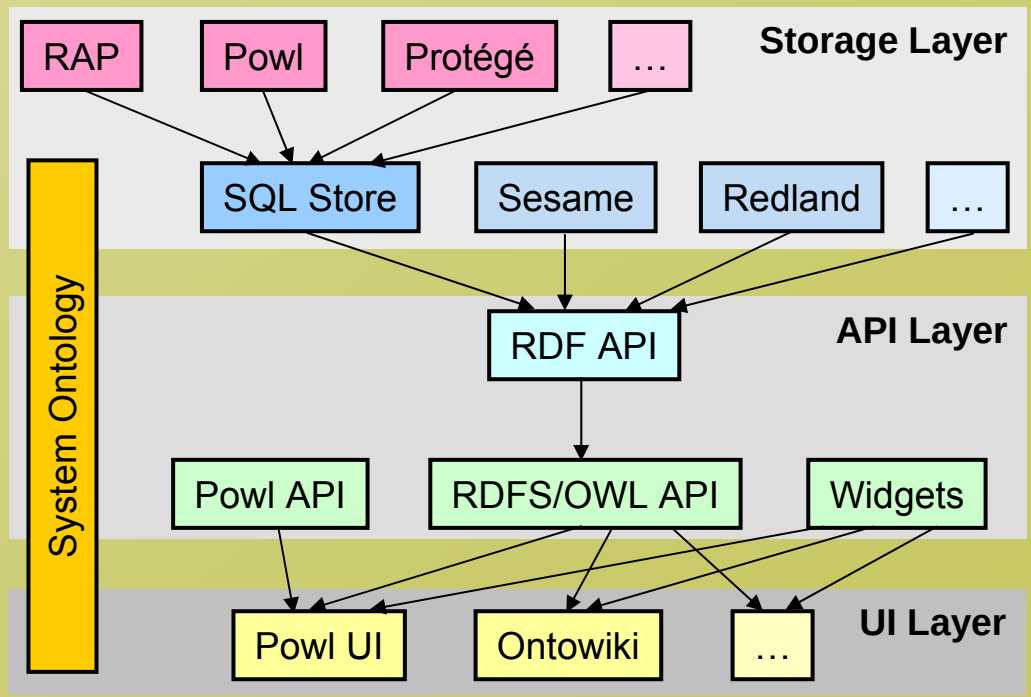
- Supports different views on a KB: e.g. Triples, Serialization, DB like
- May and is used as a **foundational framework for Semantic Web applications**.
- http://powl.sf.net

Architecture

- ***Powl store*** – SQL compatible relational database backend
- ***RDFAPI (RAP), RDFSAPI, OWLAPI*** – layered APIs for handling RDF, RDF-Schema (RDFS) and OWL
- ***Powl API*** – containing classes and functions to build web applications on top of those APIs

- ***User interface*** – a set of PHP pages combining widgets accessing (browsing, viewing, editing) model data in a Powl store.

- Supports Versioning & Caching



UNIVERSITÄT LEIPZIG

Agenda

1. RapidOWL: Agile knowledge engineering methodology
2. Enabling agility: Ontology evolution and data migration
3. RDF query caching: Accelerate querying and aggregation
4. Powl: framework for Semantic Web application development
5. **OntoWiki: Semantic Web application supporting RapidOWL**
6. “Vernetzte Kirche“: RapidOWL use case

OntoWiki – Supporting Agile KE

Close in the spirit to existing Wiki systems

Triples not texts: technologically independent & complementary to Wikis

Differs from strategies to integrate Wiki systems and SW (embed triples into Wiki texts in special syntax)

Obstacles:

- **Usability:** more and more syntactic possibilities counteracts ease of use
- **Redundancy:** answering real-time queries requires triple store
- **Scalability:** changes involving statements with different subjects don't scale

Apply Wiki paradigm: **``making it easy to correct mistakes, rather than making it hard to make them"**.

Rapidly simplify instance data presentation & acquisition from & for end users.

View Ontologies as "information maps“:

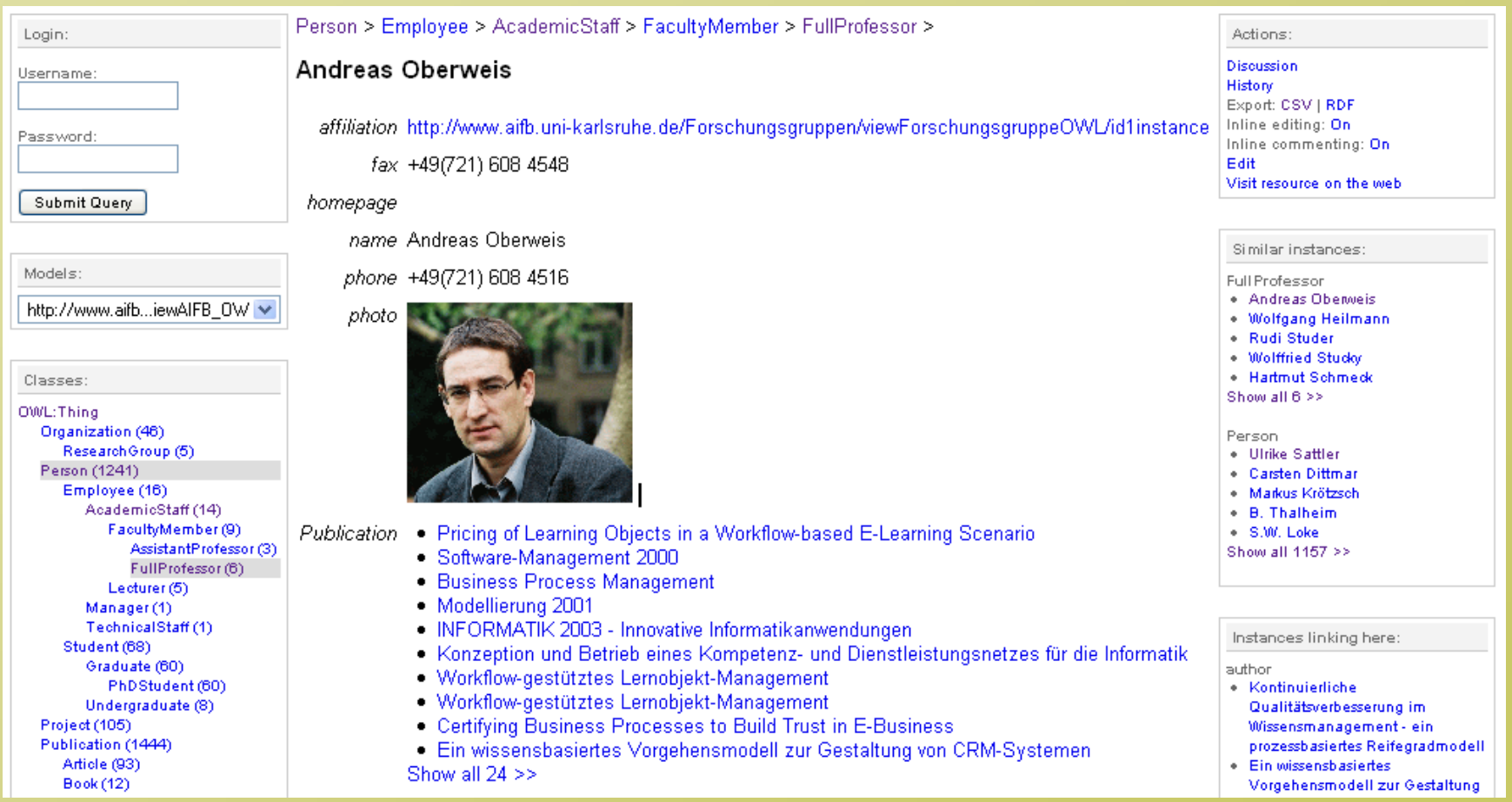
- each node of the information map is intuitively visually represented in a generic but configurable way & interlinked to related digital resources.
- enable incremental enhancement of knowledge schema as well as the instance data

Requirements

- **Intuitive display and editing** of instance data,
- **Semantic views**, allow generation of different views and aggregations
- **Versioning and Evolution**, tracking of changes for review & selective rolled-back, transformation of structure & data
- **Semantic search**, easy to use full-text search on all literal data, search results will be filtered and sorted semantically
- **Community support**, discussions and comments on smallest possible information chunks, votings about correctness
- **Online statistics**, interactively measure popularity of content
- **Reasoning,** Ensuring terminological consistency

UNIVERSITÄT LEIPZIG

OntoWiki Prototype



Display: RDF Templates

Design Goals:

- easy to learn and use,
- focused on text-oriented RDF, RDF-S and OWL display paradigms,
- suitable for rendering different output formats like XML dialects (such as HTML, or SVG), plain text formats (such as BibTeX), and others,
- built on existing Web technology, and
- extensible for more specialized needs.

```
<rwt:template language="SPARQL"
    query="SELECT ?book,?title WHERE (?book <dc:title> ?title)">

   <a href="{book|plain}">{title}</a><br />

</rwt:template>
```

UNIVERSITÄT LEIPZIG

OntoWiki Authoring

swrc:FullProfessor >

Andreas Oberweis

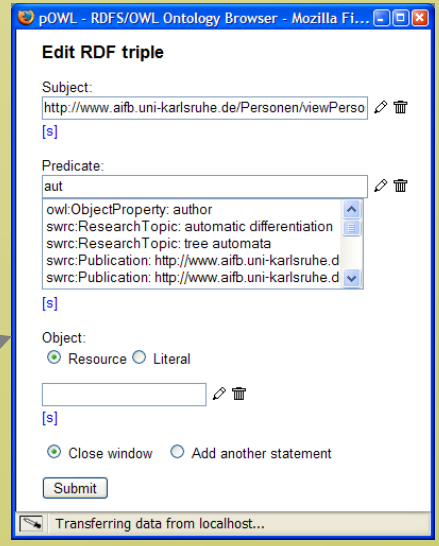
swrc:name Andreas Oberweis

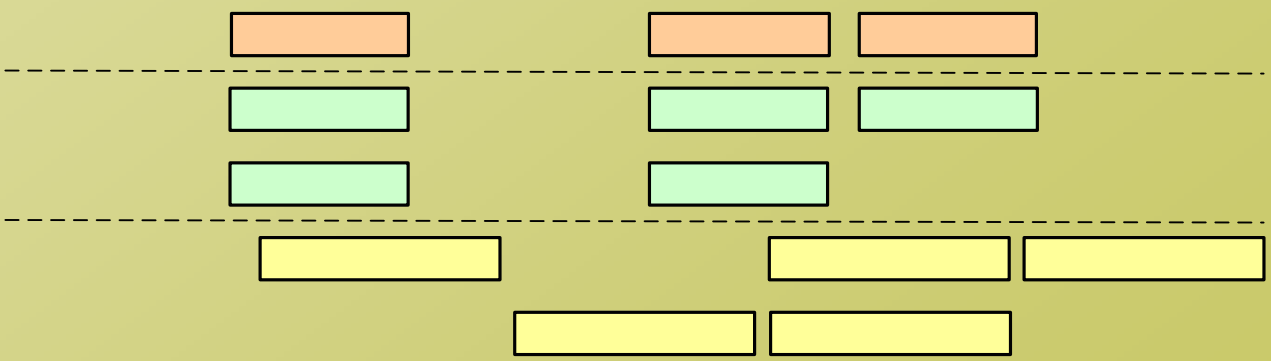
swrc:phone +49 (721) 608 4516

swrc:photo 

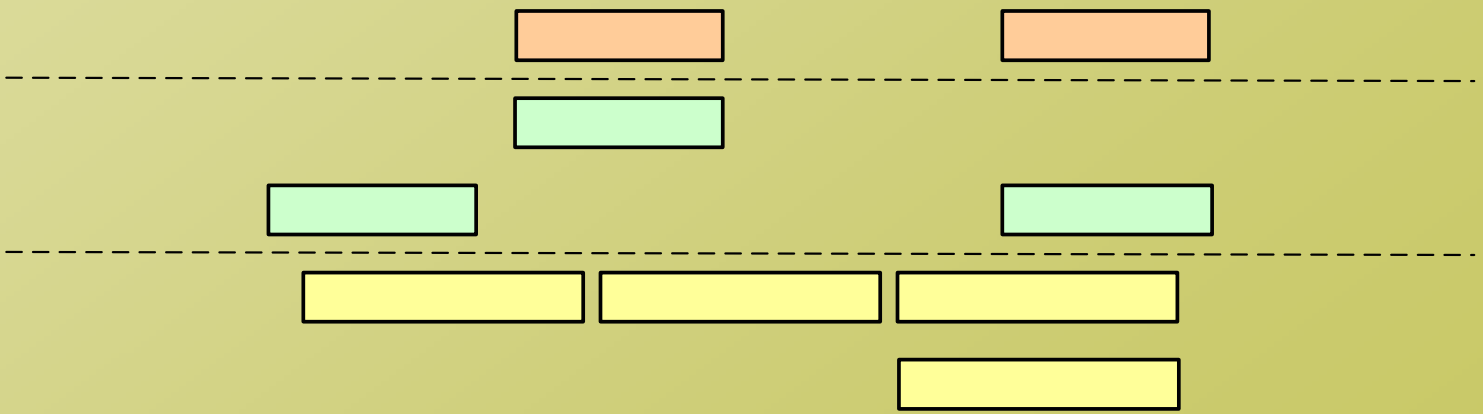
swrc:publication
- Wissensmanagement im CRM
- Flexibilität in betrieblichen Informationssystemen
- Information Technology Practitioner Skills in Europe: Current Status and Challenges for the Future

Show all 27 >>





Community Collaboration



Semantic Search

Search

Filter: All classes | All properties | Submit Query

All properties
swrc:address (2)
swrc:booktitle (3)
swrc:name (1)

Search returned **6** results.[c]

Relevance	Resource	Property	Value
100%	AssistantProfessor: York Sure	name	York Sure
100%	InProceedings: Studies on the Dynamics of Ant Colony Optimization Algorithms	booktitle	Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002), New York
100%	Proceedings: Business Process Management	address	Berlin, Heidelberg, New York
100%	InProceedings: REMINDIN': Semantic Query Routing in Peer-to-Peer Networks Based on Social Metaphors	address	New York, USA
100%	InProceedings: Developing and Managing Software Components in an Ontology-based Application Server	booktitle	Proceedings of the WWW2004 Workshop on Application Design, Development and Implementation Issues in the Semantic Web, New York, NY, USA, May 18, 2004
100%	InProceedings: Reuse Of Problem-Solving Methods In Knowledge Engineering (short paper)	booktitle	Proceedings of the 6th Annual Workshop on Software Reuse (WISR-6), Owego, New York, November 1-4, 1993

UNIVERSITÄT LEIPZIG

Agenda

1. RapidOWL: Agile knowledge engineering methodology
2. Enabling agility: Ontology evolution and data migration
3. RDF query caching: Accelerate querying and aggregation
4. Powl: framework for Semantic Web application development
5. OntoWiki: Semantic Web application supporting RapidOWL
6. **“Vernetzte Kirche“: RapidOWL use case**

Initial Situation

!

Lutheran Church in Bavaria:

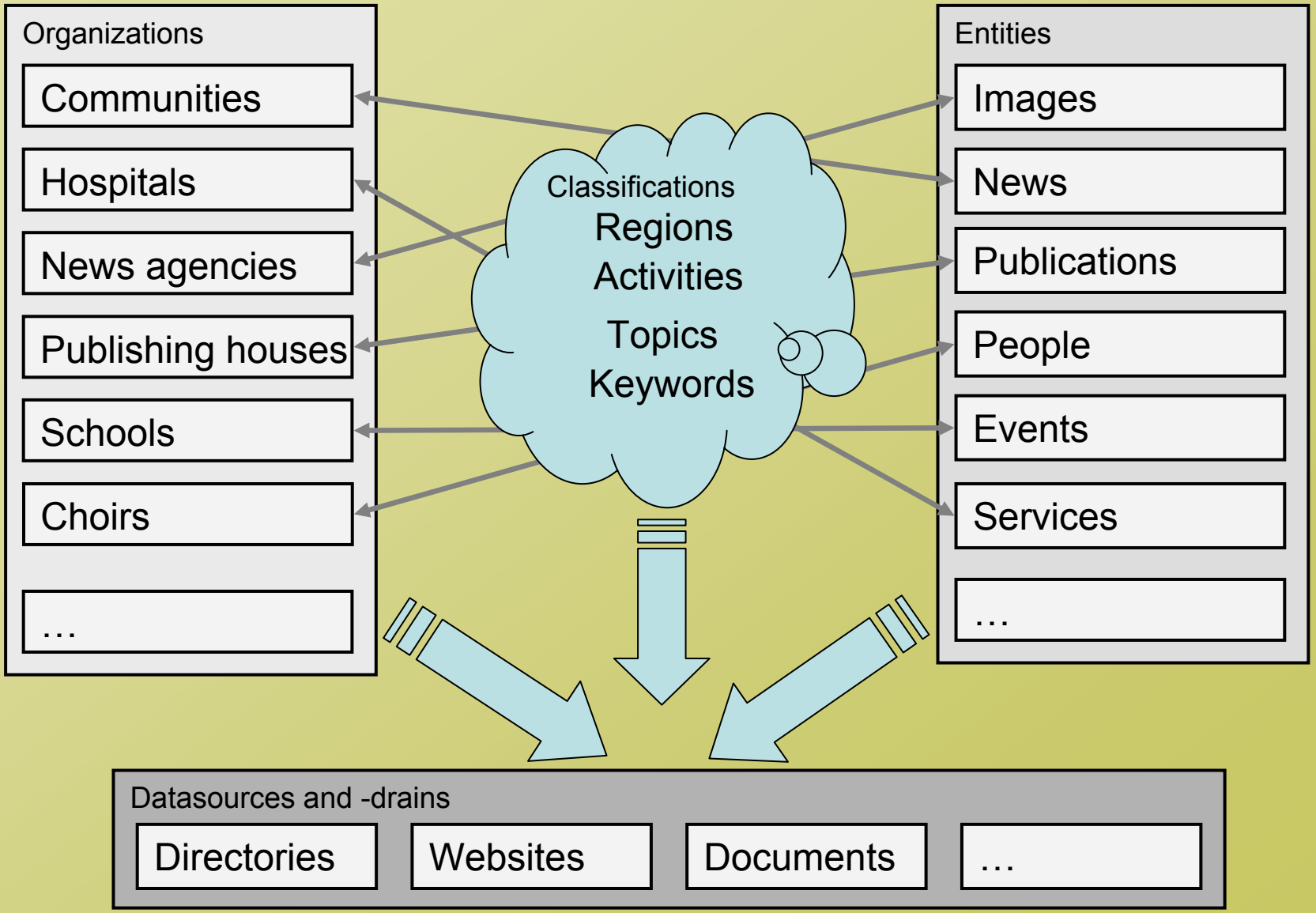
- ~2000 affiliated clerical, cultural or charitable communities, organizations, societies
- separate organizational structure
- different spheres of activities, websites, contact information …
- low budget, few resources

?

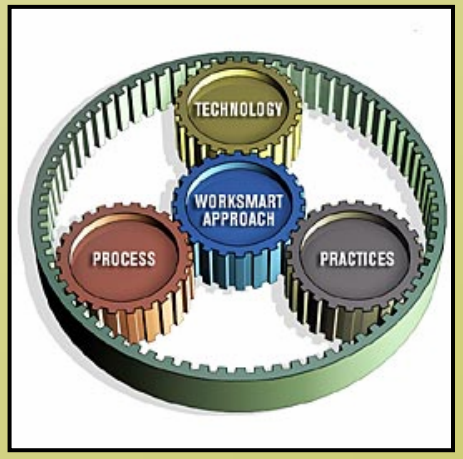
Problem:

- Preserve diversity
- Final usage scenarios not precisely defined
- Integrate information according to various classifications
- Web portal interlinking all these resources
- Easy info access in different ways for humans & machines

Information Integration Problem



Solution



RapidOWL

Semantic Web application

- collect information and meta data from distributed sources (diverse ownerships),
- integrate information into common, extensible conceptual model
- representation on the Web, exposing as many semantic relations as possible

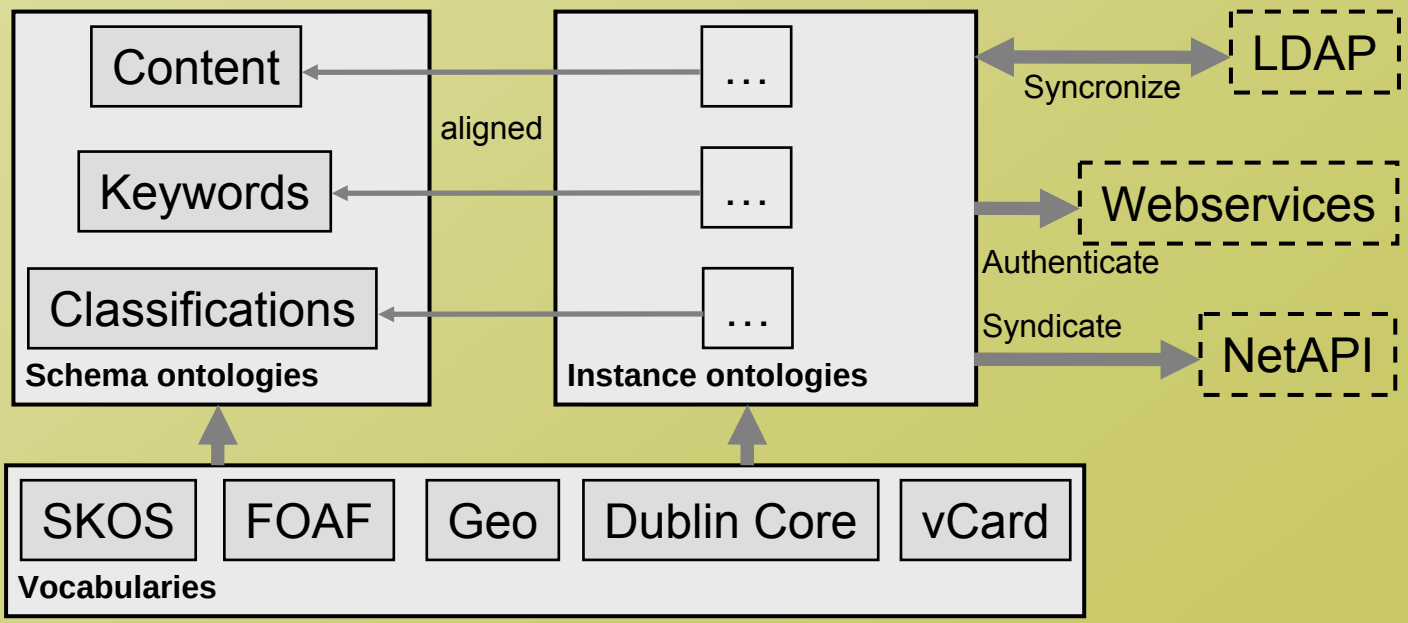
Metadata initiative

- focused on religious related content
- support individuals and organizations accordingly with its application.

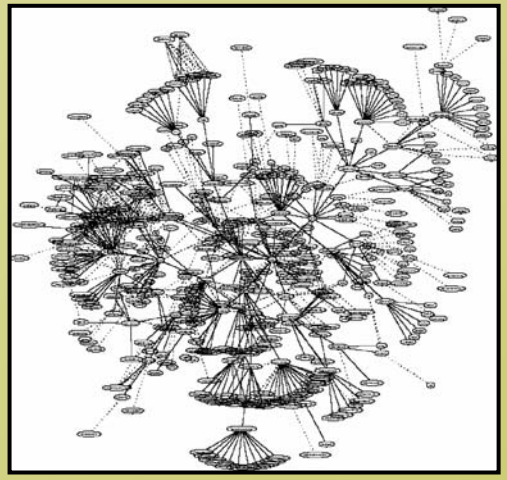
Structuring Information

Challenge:

- find integrated conceptual model for the information, flexible enough to allow integration of new information



Ontology



Main concepts (or classes):

- Organizations
- Content

  Images, Books, News, Events, Journals

Organizations - classified along **different taxonomies**:

- *Regional*,
  - By German administrative units:
    federal state, district, region, community/city
  - By established regional church structure:
    regional church, deanery, clerical district, religious community
- *Sphere of activity*

  education, arts & culture, administration, souls care
- *Type of organization*

  choir, community, publishing house
- *Topics and keywords*

  semantic bible references

Ontology cont.

SKOS methodology used for keyword taxonomies
- inspired by IPTC subject reference codes
- consistent annotation (e.g. FotoBayern, RSS feeds)

Address information + OpenGeoDB.de data = geo coordinates
- relate instances to regional classifications
- display maps

Initial acquisition of instance data:
- **import** rows **from Excel** sheets as instances
- customized generation CSV reports replaces Excel sheets

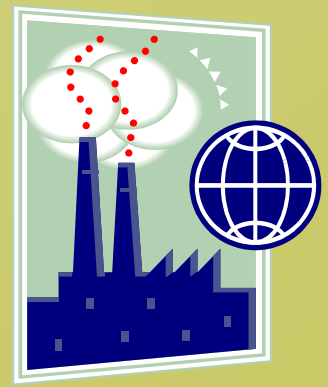
Ontology schema:
- Developed using Powl
- **available** at http://www.vernetzte-kirche.de
- **reusable** by other established regional churches or similar federally structured associations - **Virtual Organizations**
- 70 classes and 40 properties

Continuous Information Acquisition

Continuously acquire, synchronize, update instance data

For content related metadata, specialized acquisition methods:

- **Images** – metadata integrated according to VK classifications.
  - Save metadata within the file.
  - Gather images and metadata from specified web sources (FotoBayern)
- **News** – e.g. http://sonntagsblatt.de or http://epd.de
  - syndicated by accessing RSS feeds.
  - Feeds shall use VK or IPTC classifications.
- **Events** – e.g. concerts, religious services or courses
  - Specialized database & portal: http://evangelische-termine.de
  - Synchronization between DB schema and ontology similar to D2RQ



Distributed ownership

Distributed ownership strategy for instance data about organizations:

- HTML header of organization website contains metadata
- Gathered by crawler, existing instances are updated accordingly

Meta-tag generator

- eases integration of metadata into the organization websites
- simple web form to generate an HTML header snippet
- metadata extraction implemented according to **GRDDL**
- **commonly used vocabularies** facilitate interpretation by other (search) agents (e.g. Dublin Core, vCard, FOAF, Geo)

LDAP-RDF synchronization since user and organization related information held in LDAP directory

Overall instance data: 5000 instances, 60000 triples.

Metadata Example

```
<meta name="DC.title" content="Vernetzte Kirche" />
<meta name="DC.creator" content="Bart Pieterse" />
<meta name="DC.subject" content="Semantisch Vernetzte Kirche" />
<meta name="DC.description" content="Informationen zur AG
   Vernetzte Kirche der Evangelisch-Lutherischen Kirche in
   Bayern und semantische Suche nach ELKB Resourcen" />
<meta name="DC.publisher" content="Vernetzte Kirche" />
<meta name="DC.date" content="2005-05-03" />
<meta name="DC.language" content="de" />
<meta name="DC.coverage" content="Bayern" />
<meta name="DC.rights" content="GFDL" />
<meta name="geo.lat" content="48.1333" />
<meta name="geo.long" content="-11.5833" />
<meta name="geo.country" content="BRD" />
<meta name="dmoz.id" content="" />
<meta name="keywords" content="Vernetzte Kirche, Semantic Web,
   ELKB, semantische Suche" />
<meta name="robots" content="follow" />
<meta name="revisit-after" content="20 days" />
```

Sharing information

… with “human” and artificial agents:

- machine consumption:
  Web-Services or the NetAPI (share RDF ontology (fragments) by using the HTTP protocol)
- for humans:
  bunch domain or topic specific views on the data

Powl's API used to present information on portal:
http://www.vernetzte-kirche.de.

- efficiently search and filter the instance data
- full-text search to query the data for literal descriptions

UNIVERSITÄT LEIPZIG

The Portal

Projekt Vernetzte Kirche

Nach Regierungskreisen:
Mittelfranken
Niederbayern
Oberbayern
Oberfranken
Oberpfalz
Schwaben
Unterfranken

Nach Dekanaten:
Kirchenkreis Ansbach - Würzburg
- Dekanat Ansbach
- Dekanat Aschaffenburg
- Dekanat Bad Neustadt
- Dekanat Bad Windsheim
- Dekanat Castell
- Dekanat Dinkelsbühl
- Dekanat Feuchtwangen
- Dekanat Gunzenhausen
- Dekanat Heidenheim
- Dekanat Kitzingen
- Dekanat Leutershausen
- Dekanat Lohr am Main
- Dekanat Markt Einersheim
- Dekanat Rothenburg o.T
- Dekanat Schweinfurt
- Dekanat Uffenheim
- Dekanat Wassertrüdingen
- Dekanat Windsbach
- Dekanat Würzburg

Kirchenkreis Augsburg
Kirchenkreis Bayreuth
Kirchenkreis München
Kirchenkreis Nürnberg
Kirchenkreis Regensburg

Organisationen im Web

Die Suche ergab **954** Ergebnisse.[c]
|< | ***1*** | 2 | 3 | 4 | ... | >|

Amt Amt für Gemeindedienst Nürnberg
Details Website

Diakonisches Werk Innnere Mission München
Details Website

Beratungsstelle AIDS-Beratung Mittelfranken der Stadtmission Nürnberg
Details Website

Handlungsfelder:
Spiritualität, Gottesdienst, Verkündigung und Kirchenmusik
Gemeindeaufbau und Gemeindeentwicklung
Seelsorge und Beratung
Themen- und zielgruppenbezogene gesellschaftliche Dienste
Ökumene, Mission, Entwicklungsdienst und Partnerschaft
Diakonisches Handeln
Presse, Öffentlichkeit, Medien
Aus-, Fort- und Weiterbildung

Organisationsart:
Amt
Beratungsstelle
Bildungszentrum
Chor
Diakonisches Werk
Evangelische Jugend
Hochschule
Institut
Kirchengemeinde
Materialstelle
Medienstelle
Sonstige Organisation
Stadtmission
Tagungsstätte
Verlag
Wohnheim

Suche:

Summarizing „Vernetzte Kirche“

Semantic Web knowledge representation standards (especially Powl) enabled rapid development of:

- initial version of the ontology and
- prototype of the web portal

with **minimized resource and cost demands (AGILE approach)**.

Application is **work in progress**:

- Experiences with real users showed practicability of the approach.
- transition into easy to use, scalable industry strength system: improve usability, documentation, stability of crawling
- client-side metadata annotation and extraction methods planned

Showcases **coherently integration of** Semantic Web **standards**, **vocabularies**, and **methodologies**, delivering added value to large audience of participating peers.

UNIVERSITÄT LEIPZIG

Summary

1. **RapidOWL: Agile** knowledge engineering **methodology**
2. Ontology **evolution** and data migration
3. Accelerate querying and aggregation by **RDF query caching**
4. **Powl:** framework for Semantic Web application development
5. **OntoWiki:** Semantic Web application supporting RapidOWL
6. “Vernetzte Kirche“: showcases **coherent integration of** Semantic Web **standards**, **vocabularies**, and **methodologies**, delivering added value to large audience of participating peers.