Ontology Lifecycle

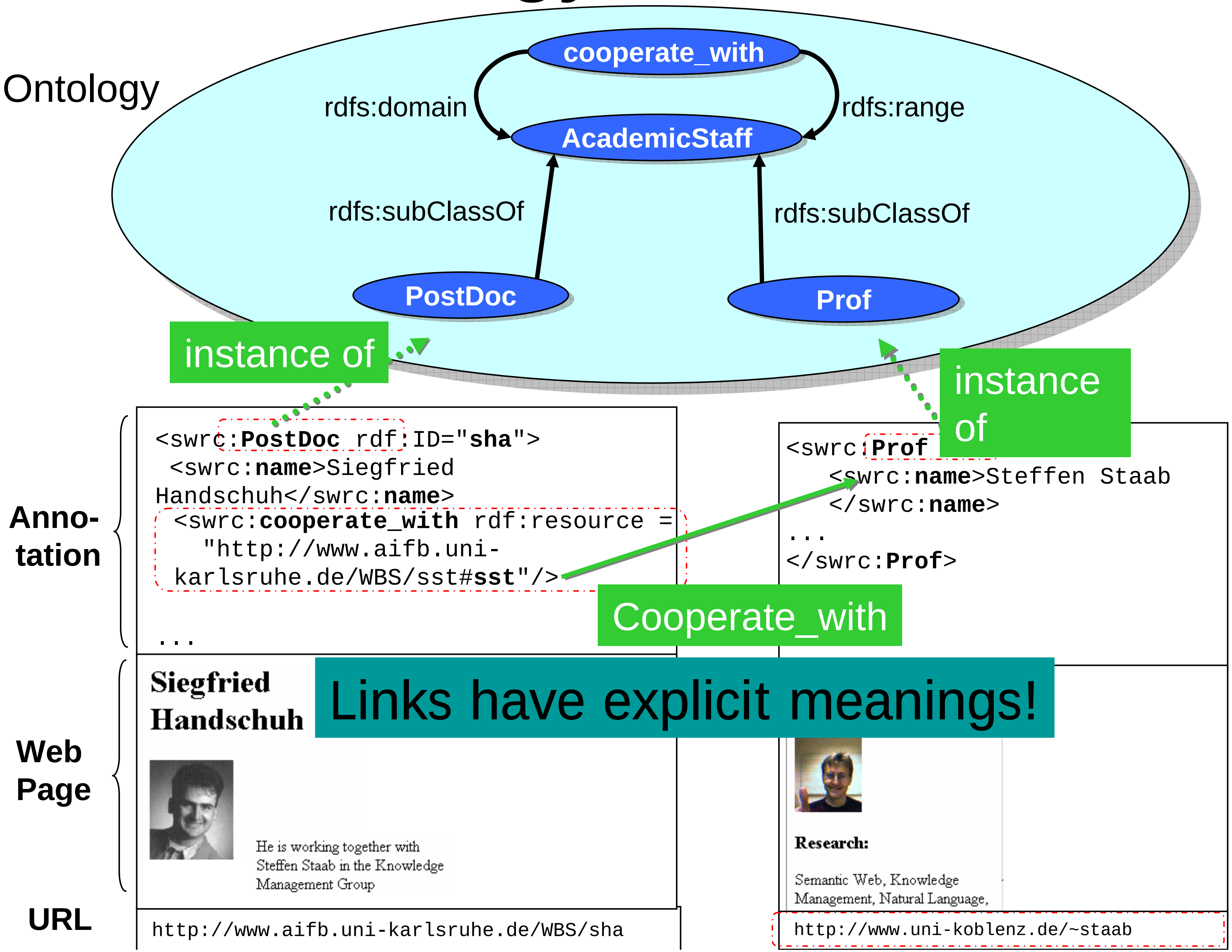
Ontology

„People can‘t **share knowledge** if they do not speak a **common language**.“ [Davenport & Prusak, 1998]

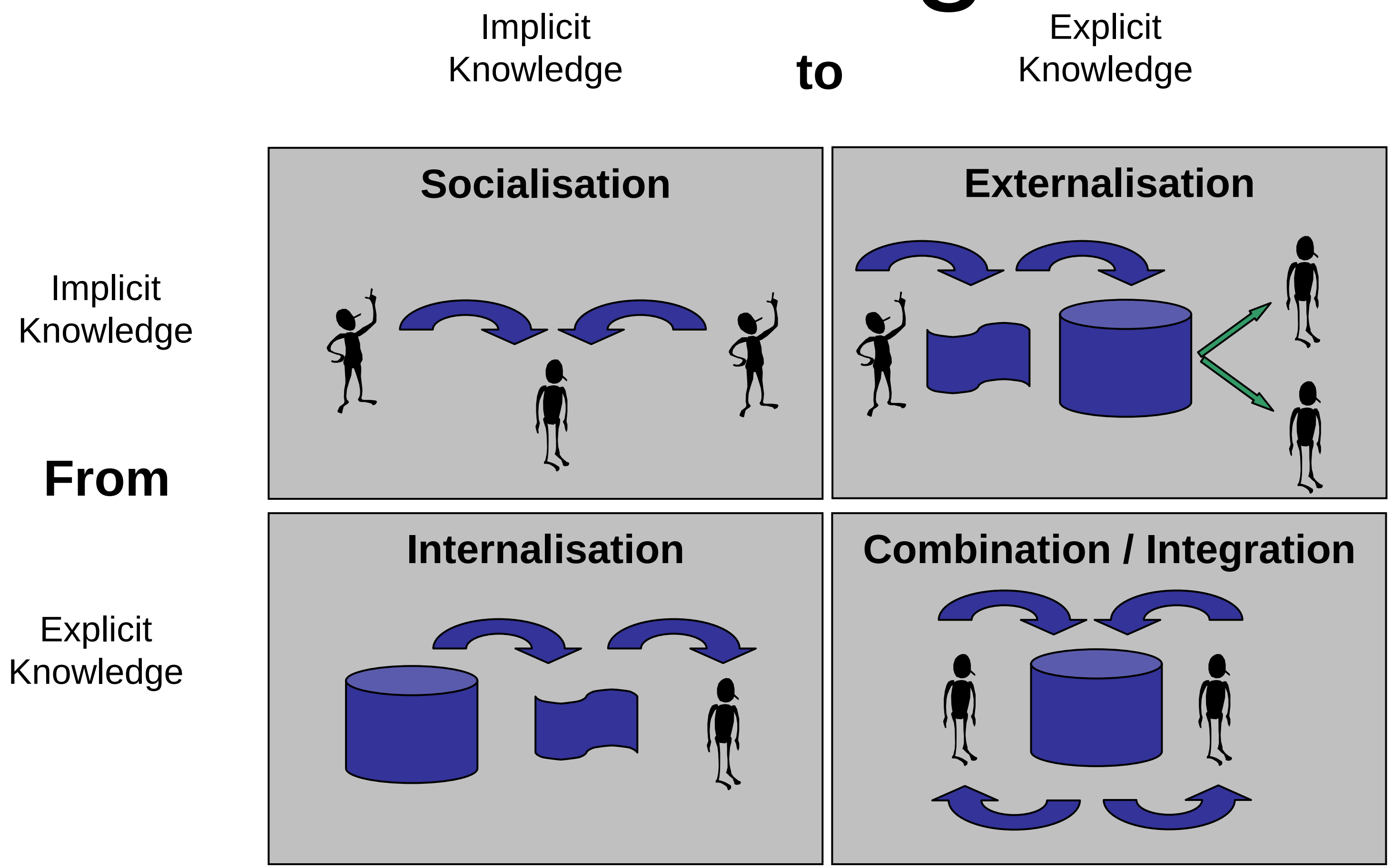
„An ontology is an **explicit specification** of a **conceptualization**.“ [Gruber, 1993]

- Ontologies enable a **better communication** between Humans/Machines
- Ontologies **standardize** and **formalize** the meaning of words through concepts

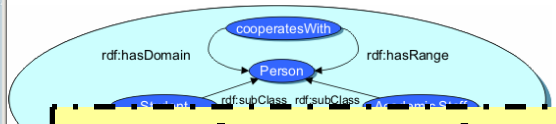
Ontology & Metadata



Explicit vs. Implicit Knowledge



Case study: OntoWeb.org

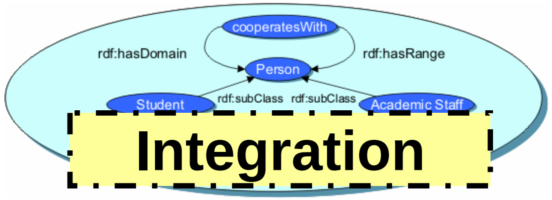


Portal Generation

Navigation

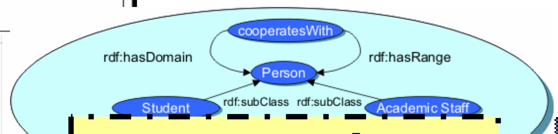
Query/Serach

Content



Integration

Collect metadata from participating partners



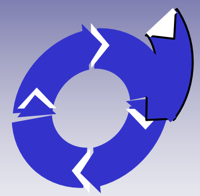
Annotation

Ontology-based Processes

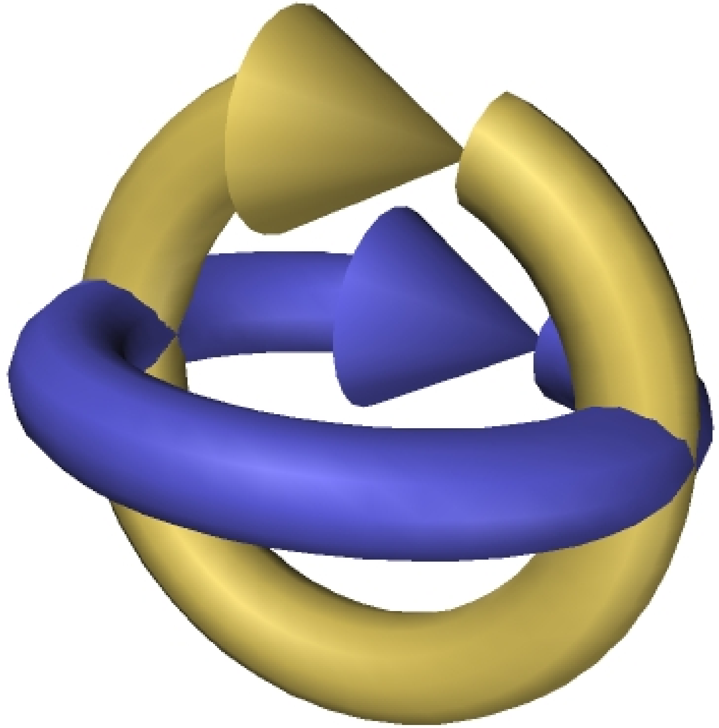
Knowledge Meta Process

Design, Implementation, Evolution of Ontology

Knowledge Process



Usage of Ontology

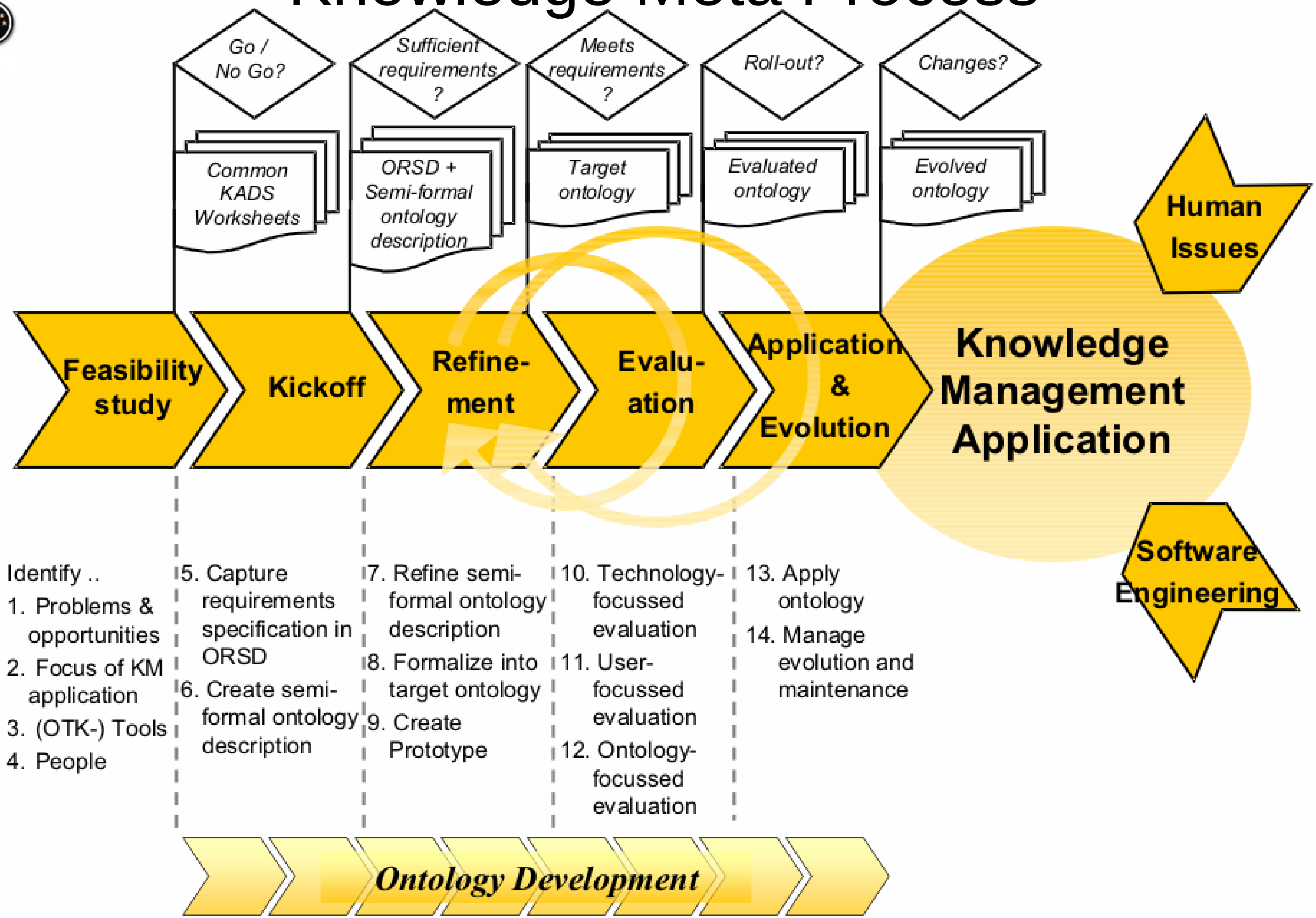


OTK Methodology:
Knowledge Meta Process

- **Task:** Build ontology based KM applications
- **Problems:**
  - **Collaboration** between domain experts and knowledge engineers
  - **Evaluation** of ontologies

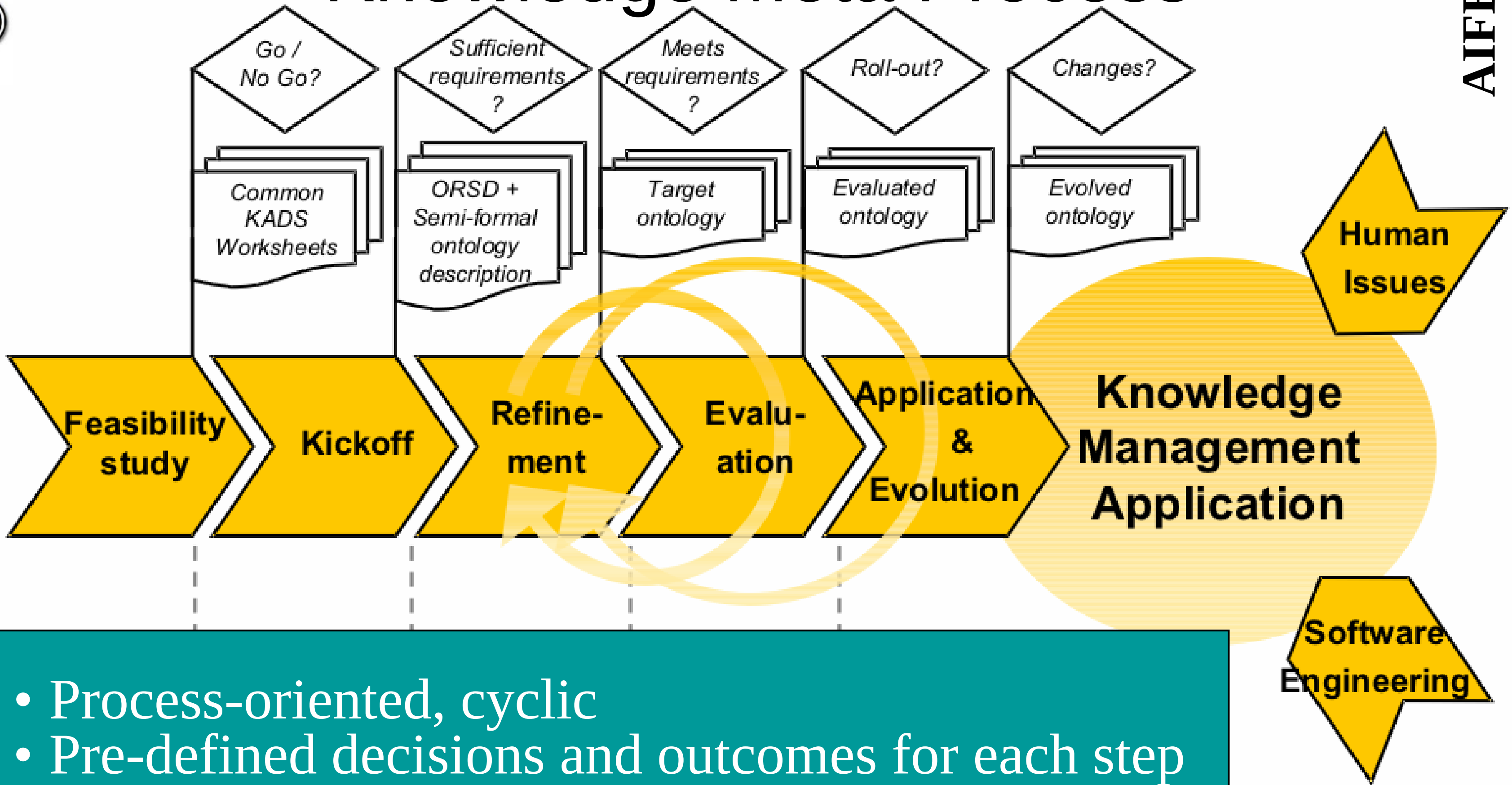
• Process-oriented, cyclic
• Pre-defined decisions and outcomes for each step
• Links to further existing methodologies for substeps

OTK Methodology:
Knowledge Meta Process



OTK Methodology:

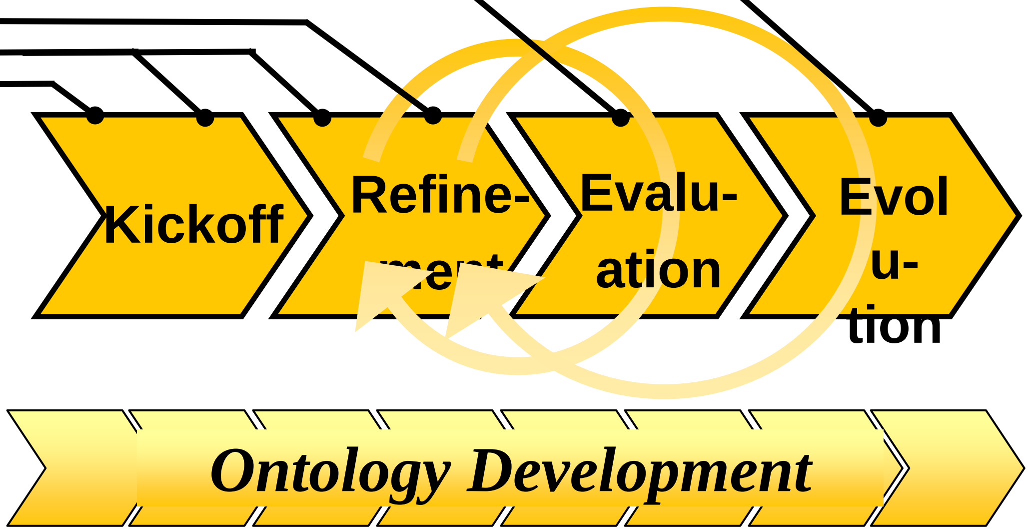
Knowledge Meta Process



- Process-oriented, cyclic
- Pre-defined decisions and outcomes for each step
- Links to further existing methodologies for substeps

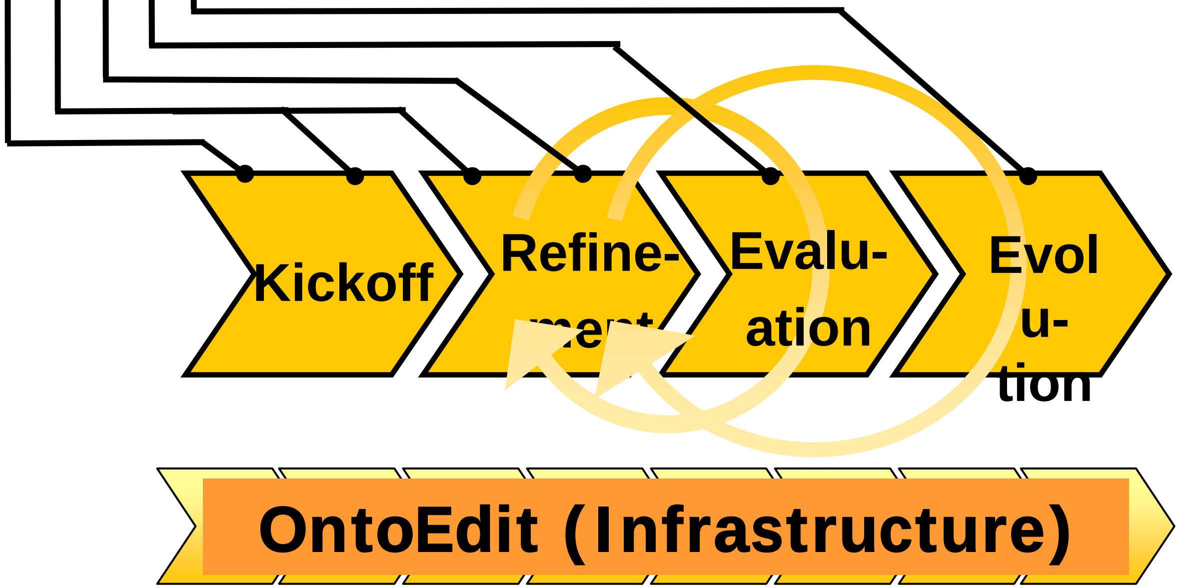
Tools

- OntoKick: Capture Requirements Specification
- Mind2Onto: Brainstorming
- OntoFiller: Documentation & Translation
- OntoClean: Formal Ontology Evaluation
- SesamePlugin: Storage & Versioning



Tools

- OntoKick: Capture Requirements Specification
- Mind2Onto: Brainstorming
- OntoFiller: Documentation & Translation
- OntoClean: Formal Ontology Evaluation
- SesamePlugin: Storage & Versioning



Feasi-
bility

Feasibility Study

- KM systems only function satisfactorily if they are properly integrated into the organization
- Many factors other than technology determine the success of such a system
- (Based on CommonKADS)

- Focus domain for ontology
- Identify people involved
- GO / No GO decision

Feasi-
bility

Feasibility study

Current State:
Skills Management

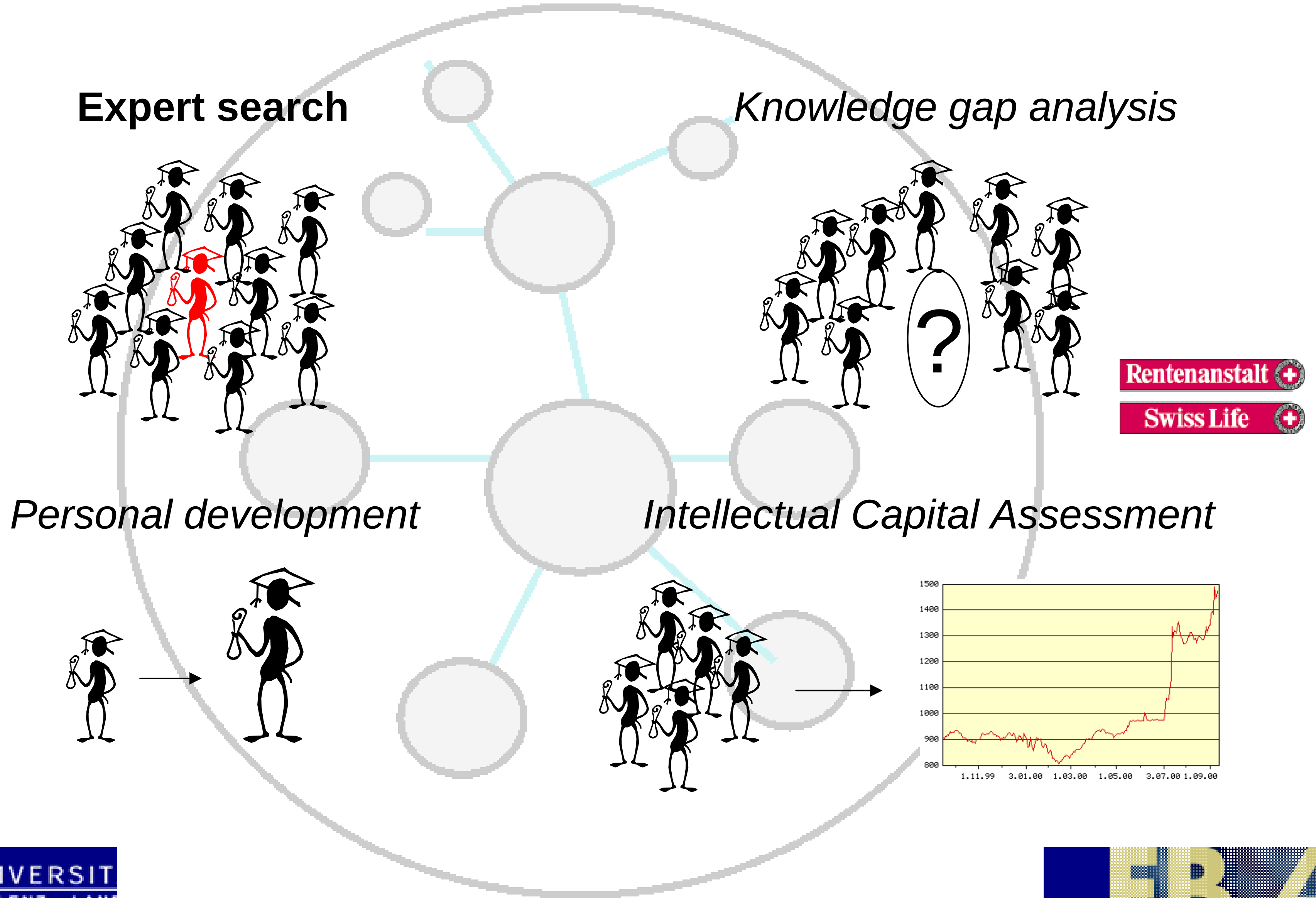
- Employee data distributed over many systems
- Different schemata for data
- Incomplete data

Rentenanstalt
Swiss Life

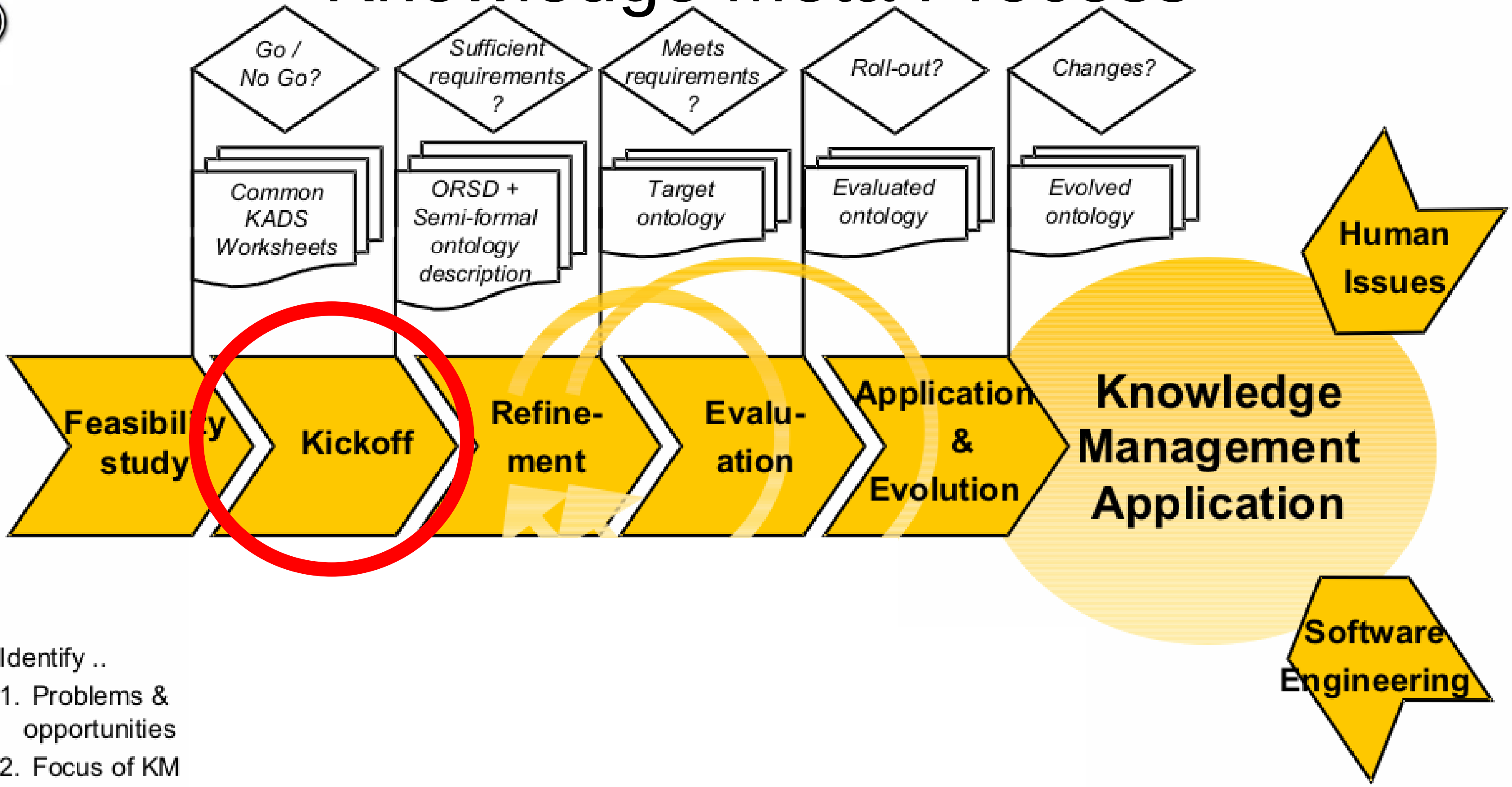
Feasi-
bility

Feasibility study

Intended state: Skills Management



OTK Methodology:
Knowledge Meta Process



Identify ..

1. Problems & opportunities
2. Focus of KM application
3. (OTK-) Tools
4. People

Kick-off

Ontology Kickoff

- Ontology Requirements Specification Document (ORSD)

  1. *Domain & Goal*
  2. *Design guidelines*
  3. *Available knowledge sources*
  4. *Potential users and user scenarios*
  5. *Applications supported by the ontolo*

  E.g. Competency questions

  Ontology Learning!

- Analyze knowledge sources
- Develop ***baseline ontology description***

  Draft version, typically most important concepts and relations are identified and described as an untyped graph

ORS – Ontology Requirements Specification

- **Goal of the ontology:**
  - Tracking and analyzing corporate business histories
- **Domain and Scope:**
  - Merger & acquisition, restructurings, management changes and other strategic activities in the chemical industry
- **Supported Applications:**
  - Web-based Corporate History Analyzer
- **Knowledge Sources:**
  - Research analysts (domain experts)
  - Document: c:/mydocuments/superdokument.doc
  - URL: http://www.webpage.com
- **Users and Use Cases:**
  - Users: Research analysts, strategic consultants
  - Use Case 1: Track strategies of specific companies
  - Use Case 2: Analyze strategic moves of competitors
- **Competency Questions:**
  - Attached Competency Questionnaire
- **Potentially reusable ontologies:**
  - not known

CQ – Competency Questionnaire

CQ Nr.	Competency Question	Concepts	Relation
CQ1	What are the subsidiaries, divisions and locations of company X?	company, subsidiary, division, location	company *has* subsidiary company *has* division company *has* location
CQ2	Which companies acquired company X?	company, acquisition	company *makes* acquisition acqusition *has* buyer acqusition *has* seller
CQ3	Which companies merged in 1990 in the rubber industry?	company, merger, year, industry	company *makes* merger company *isPartOf* industry merger *happensIn* year
CQ4	Who is CEO of company X?	CEO, company,	company *has* CEO

Kick-off

Kick-Off

- Ontology workshop to train domain experts in ontology modelling for
  .. IT
  .. Private customer insurance
  .. Human Resource Management

- First version of domain ontology by expert
  – Manual development of ontology
  – Brainstorming (Mind Maps)
  – Middle-out approach

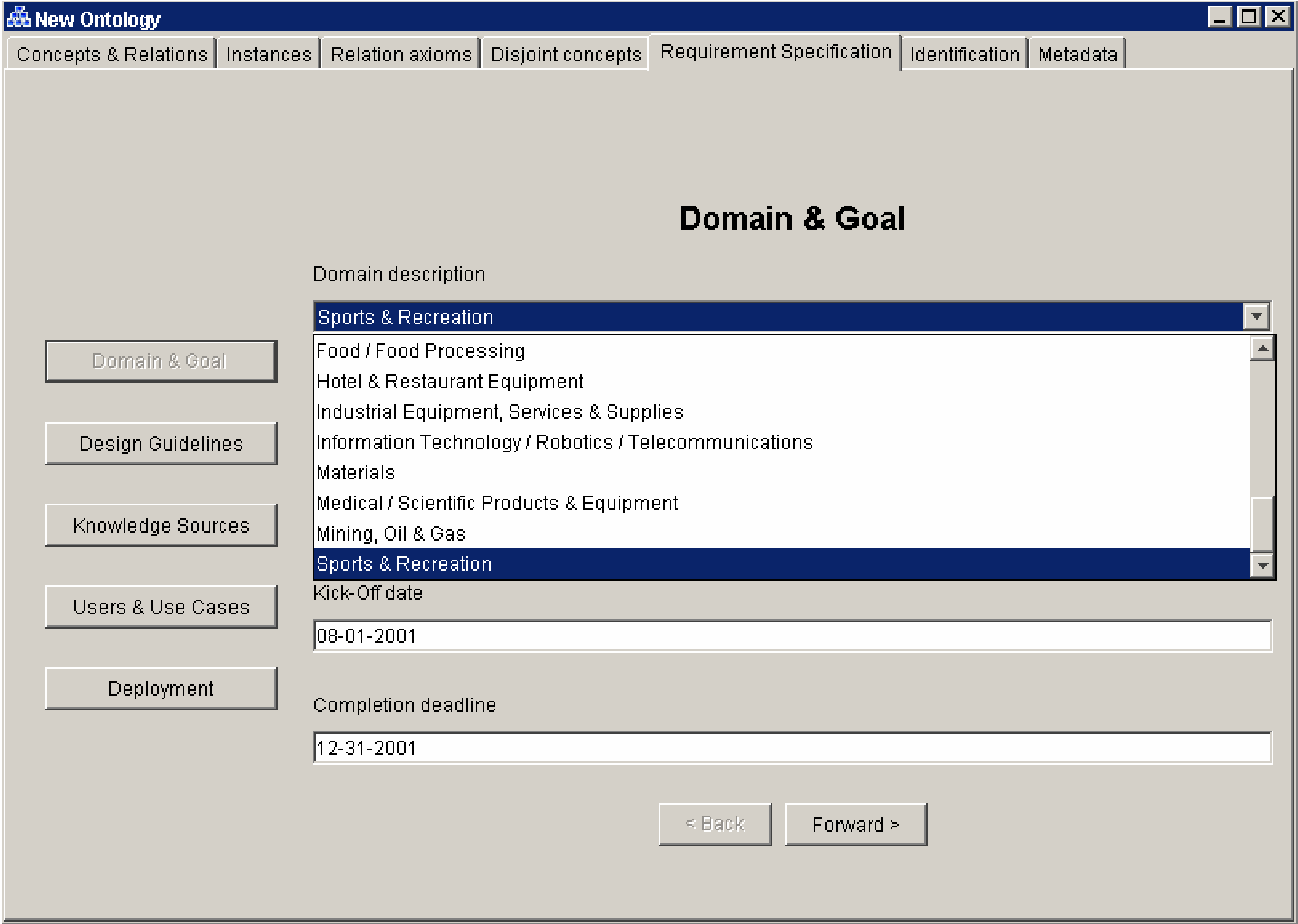
- Result: approx 700 Concepts in about 4 weeks

Kick-off

Requirement specification

Kick-off

Requirement specification

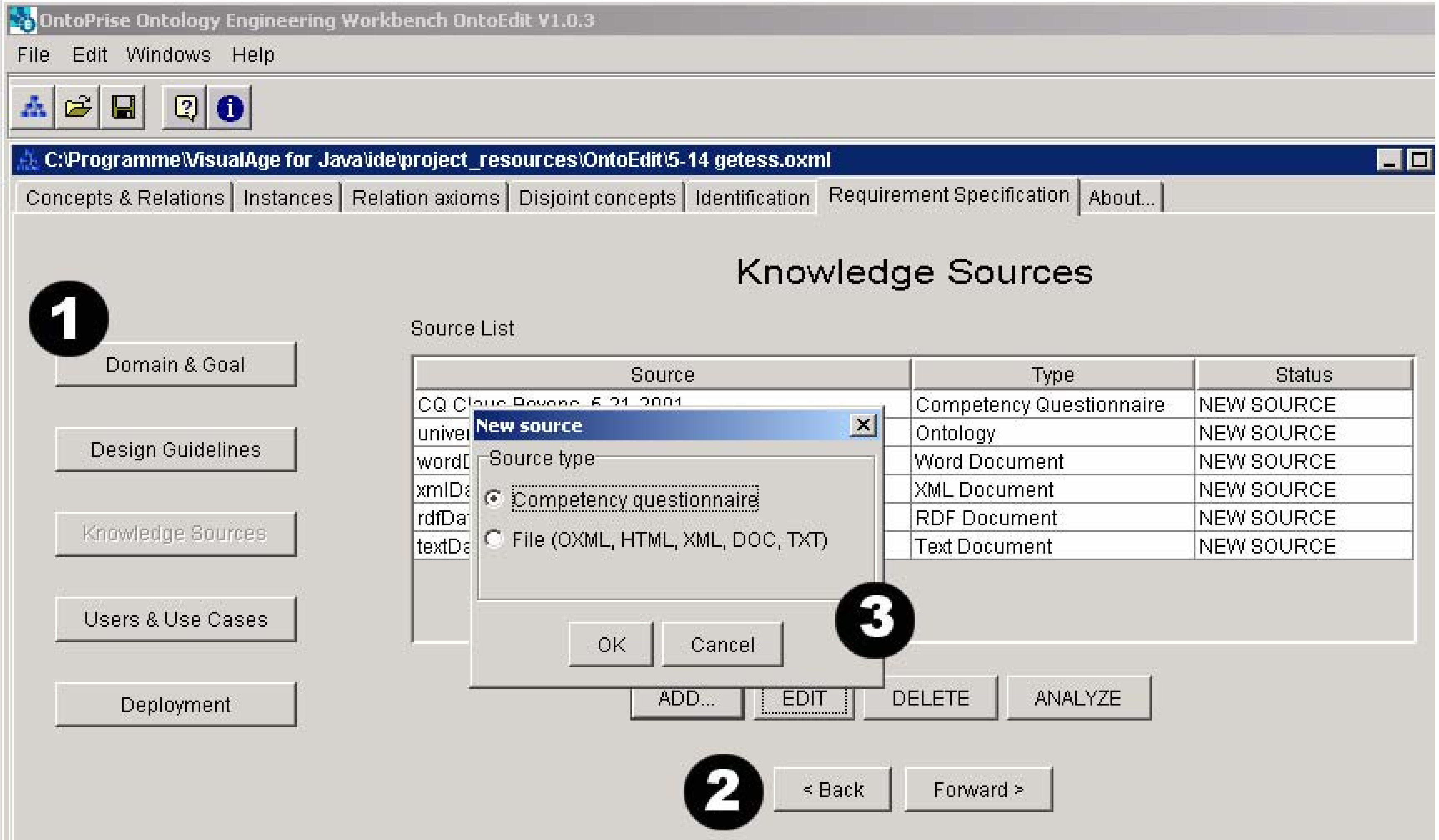


Kick-off

Requirement specification

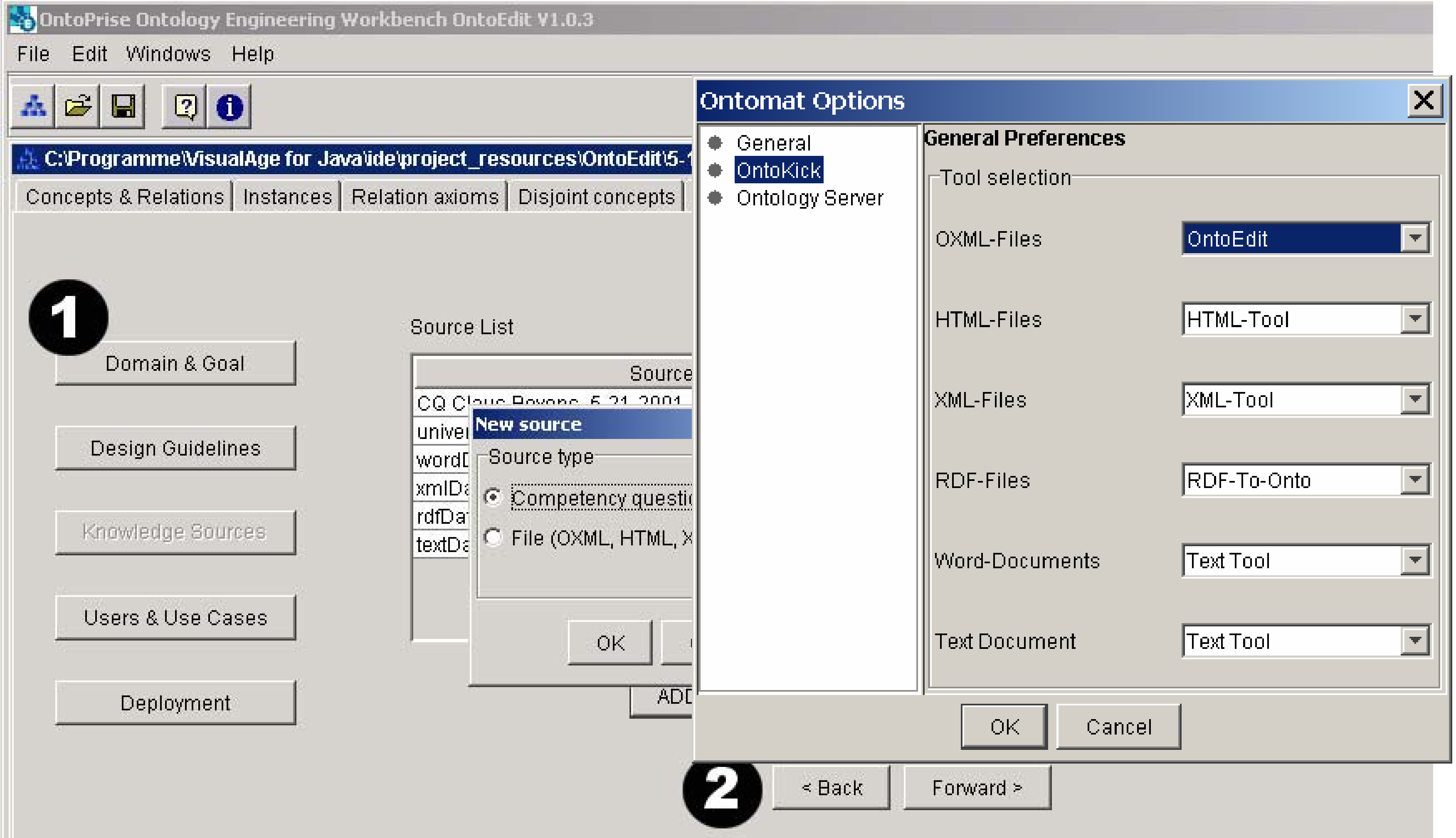
Kick-off

Knowledge Sources

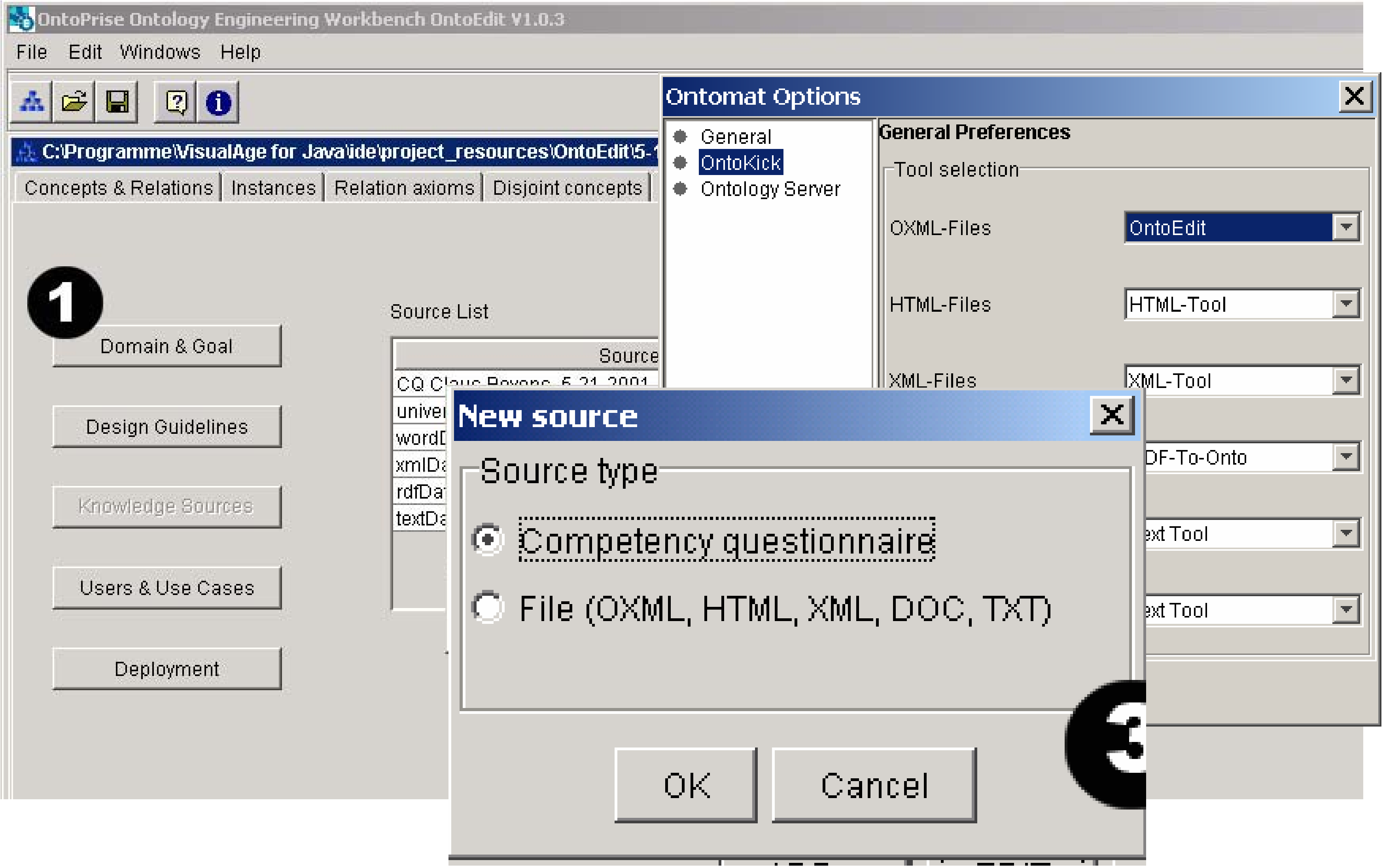


Kick-off

Knowledge Sources

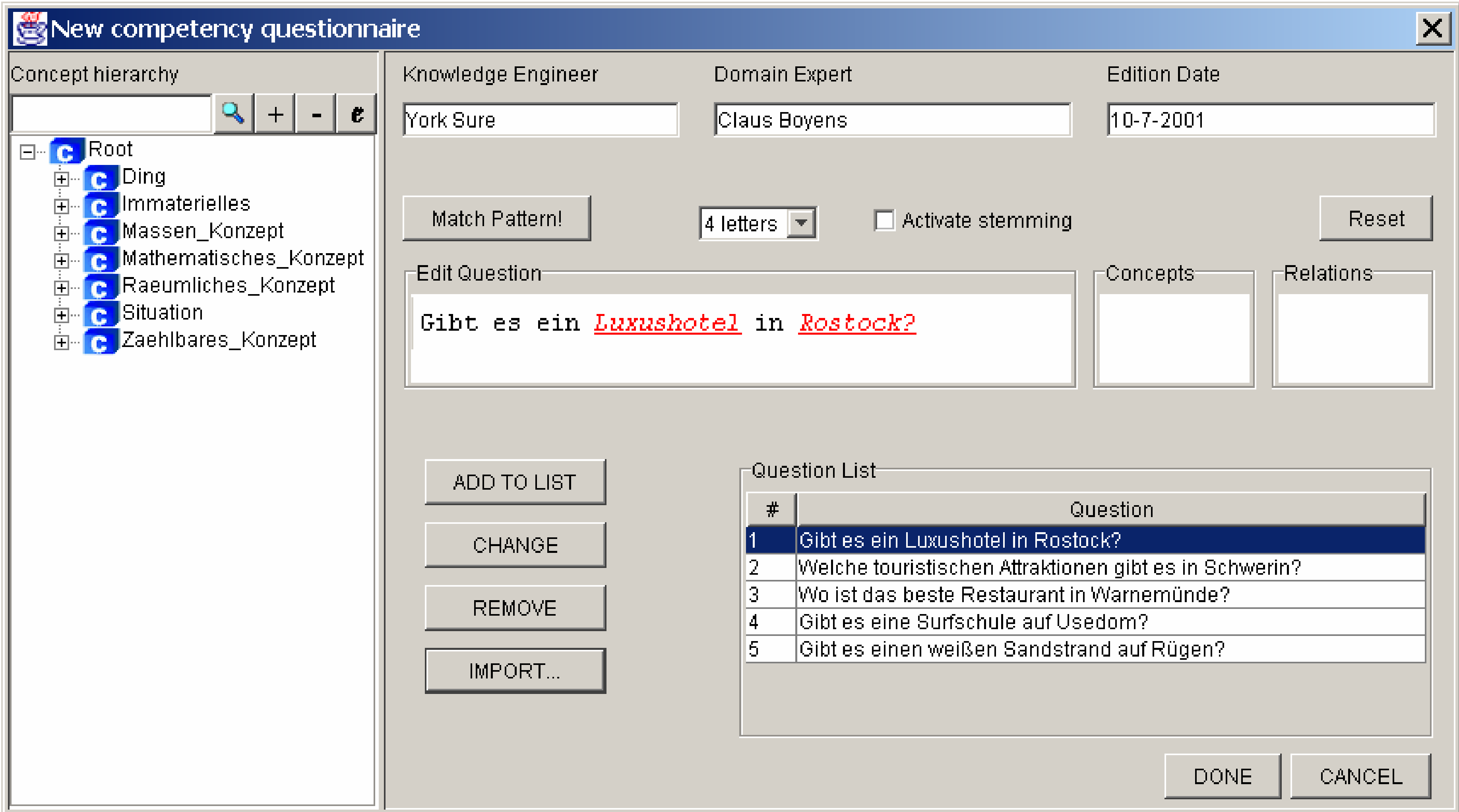


Knowledge Sources



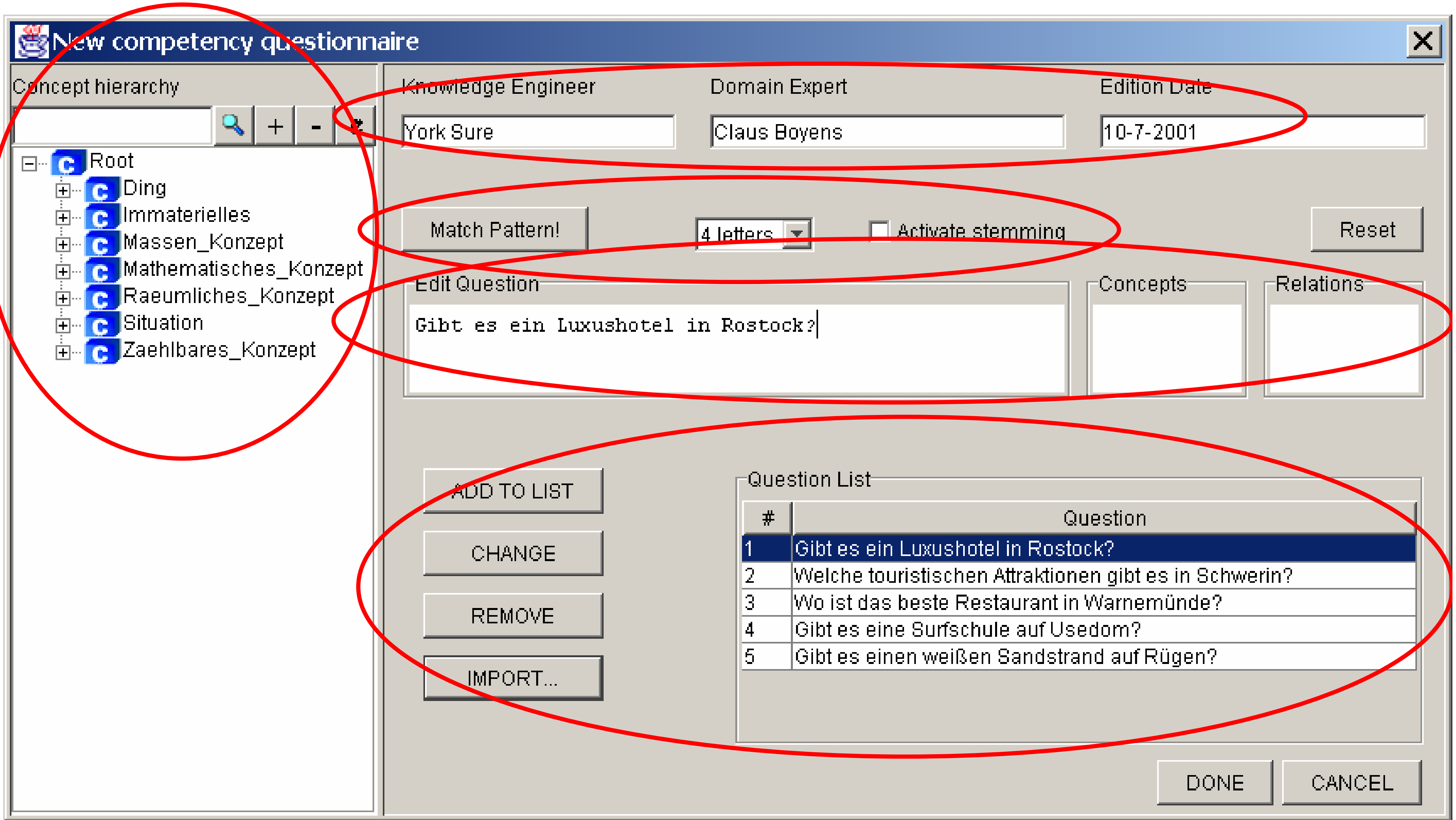
Kick-off

Competency questions



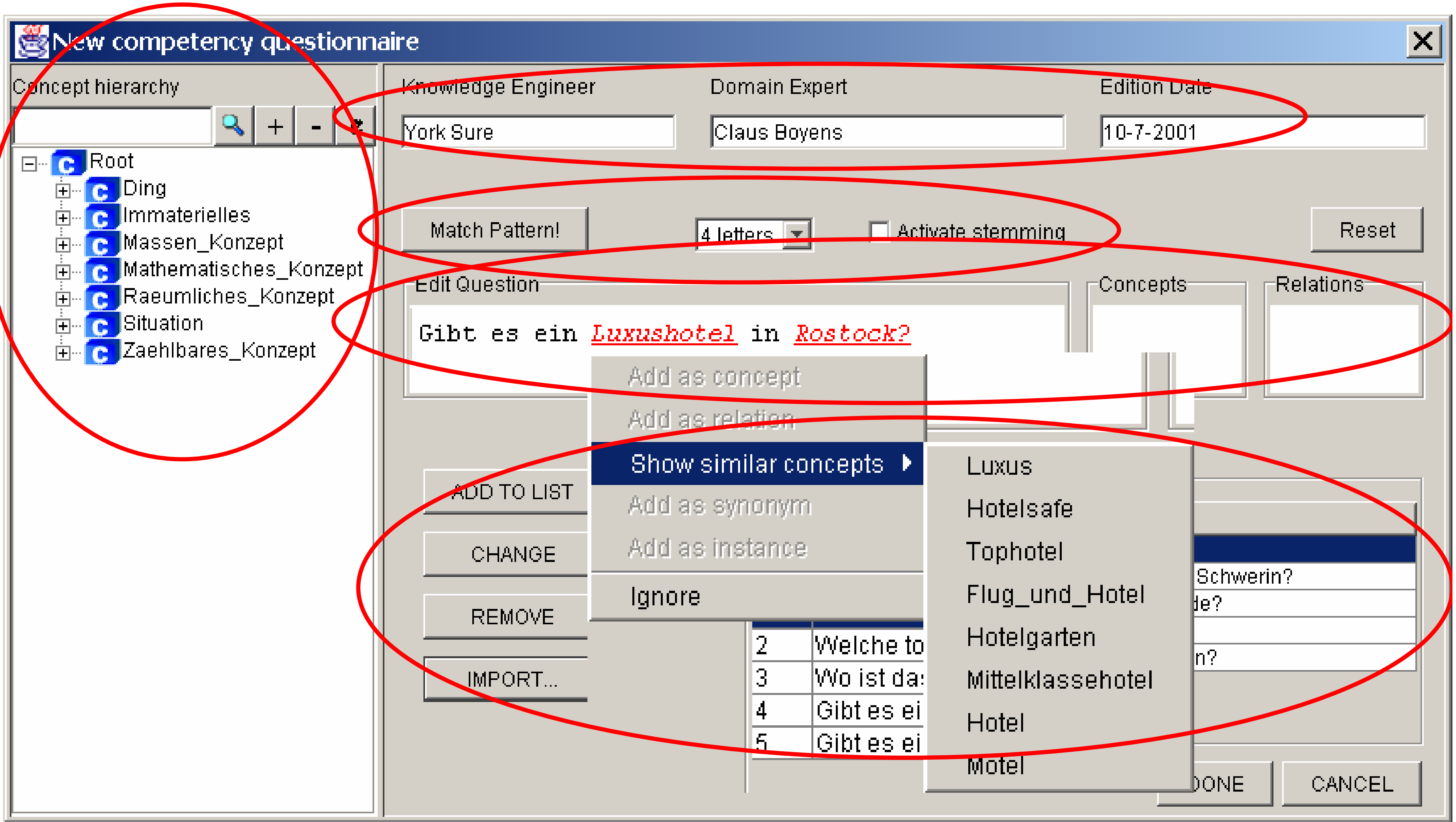
Kick-off

Competency questions



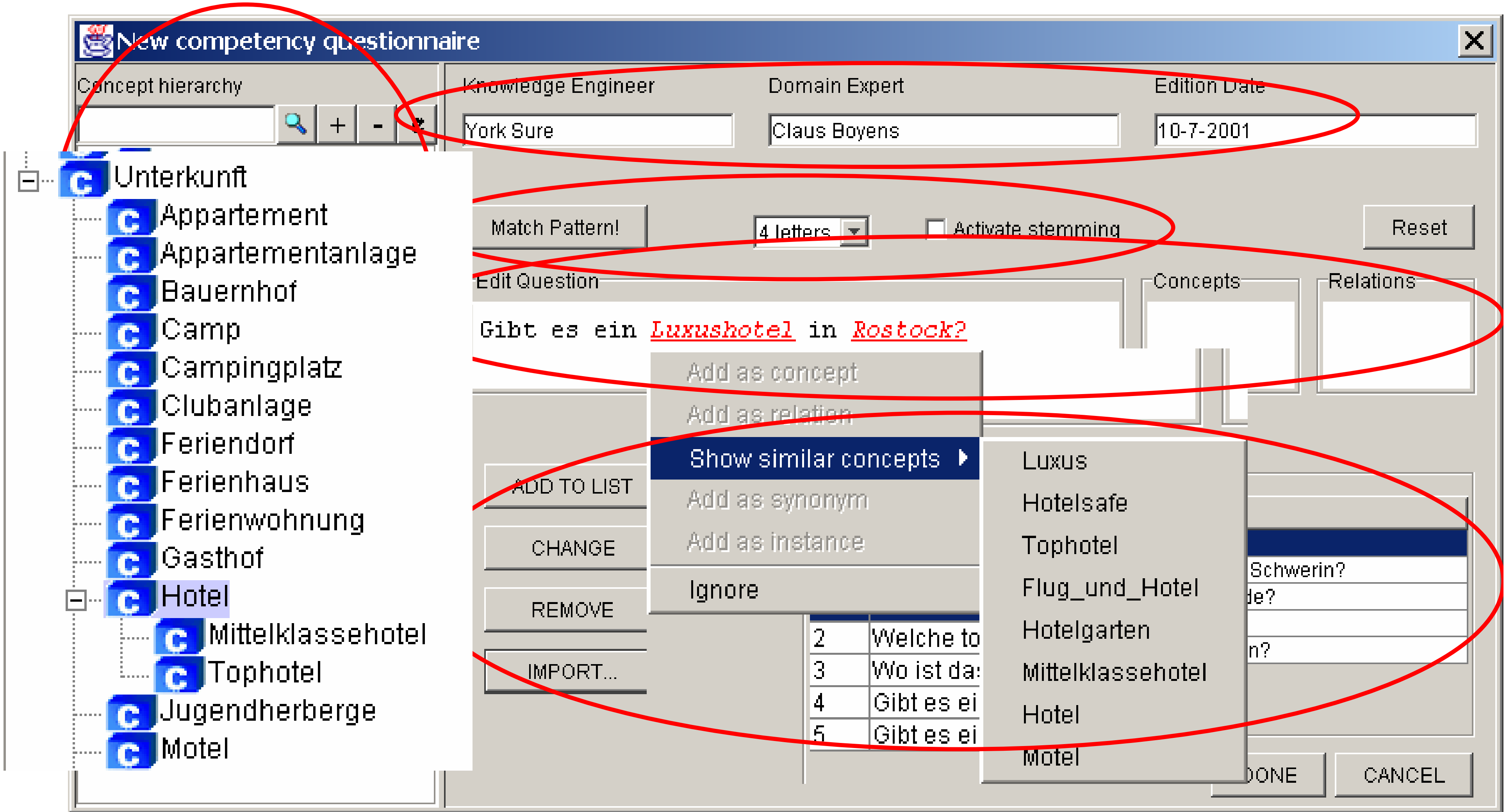
Competency questions

Kick-off



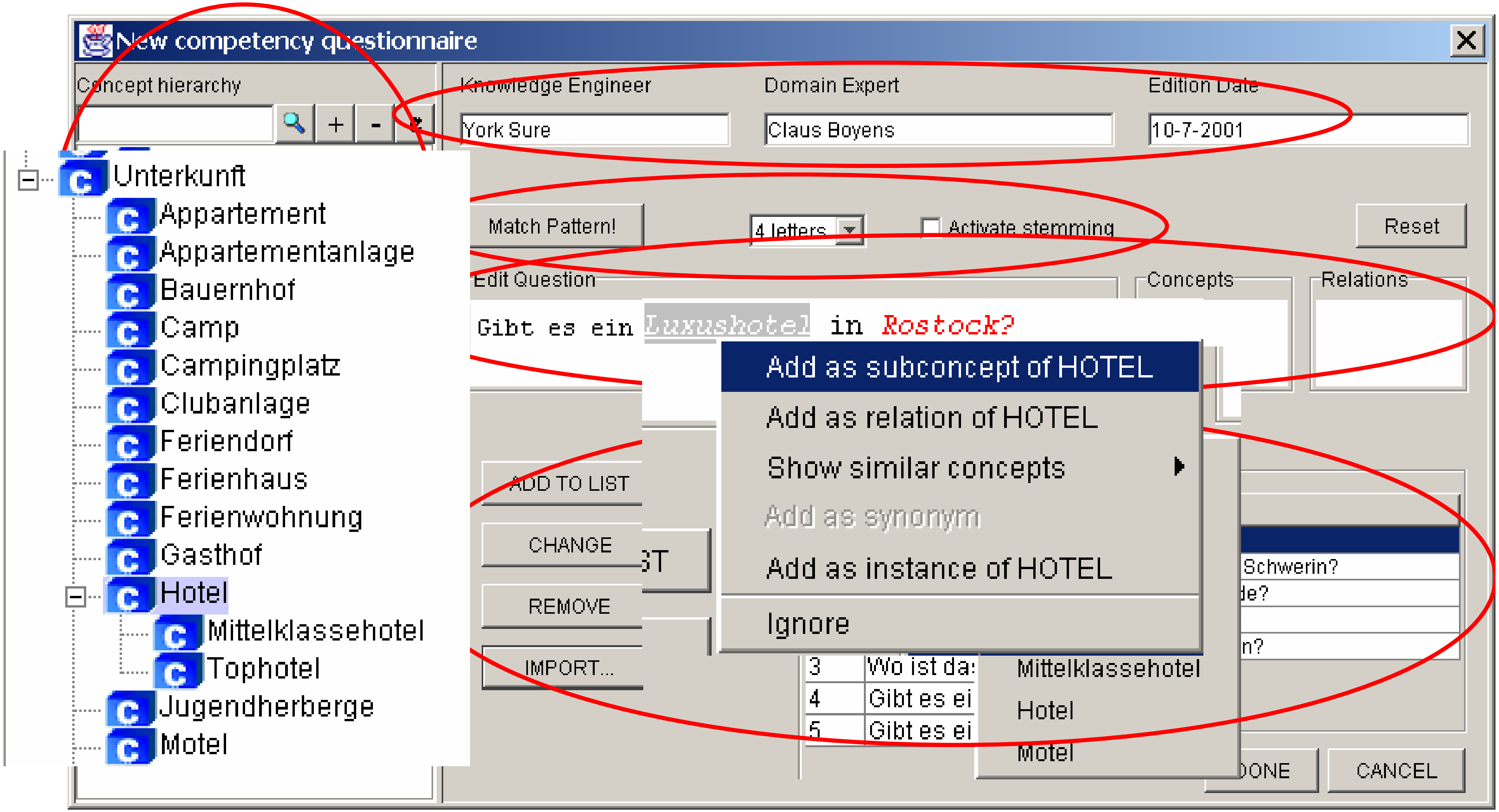
Kick-off

Competency questions



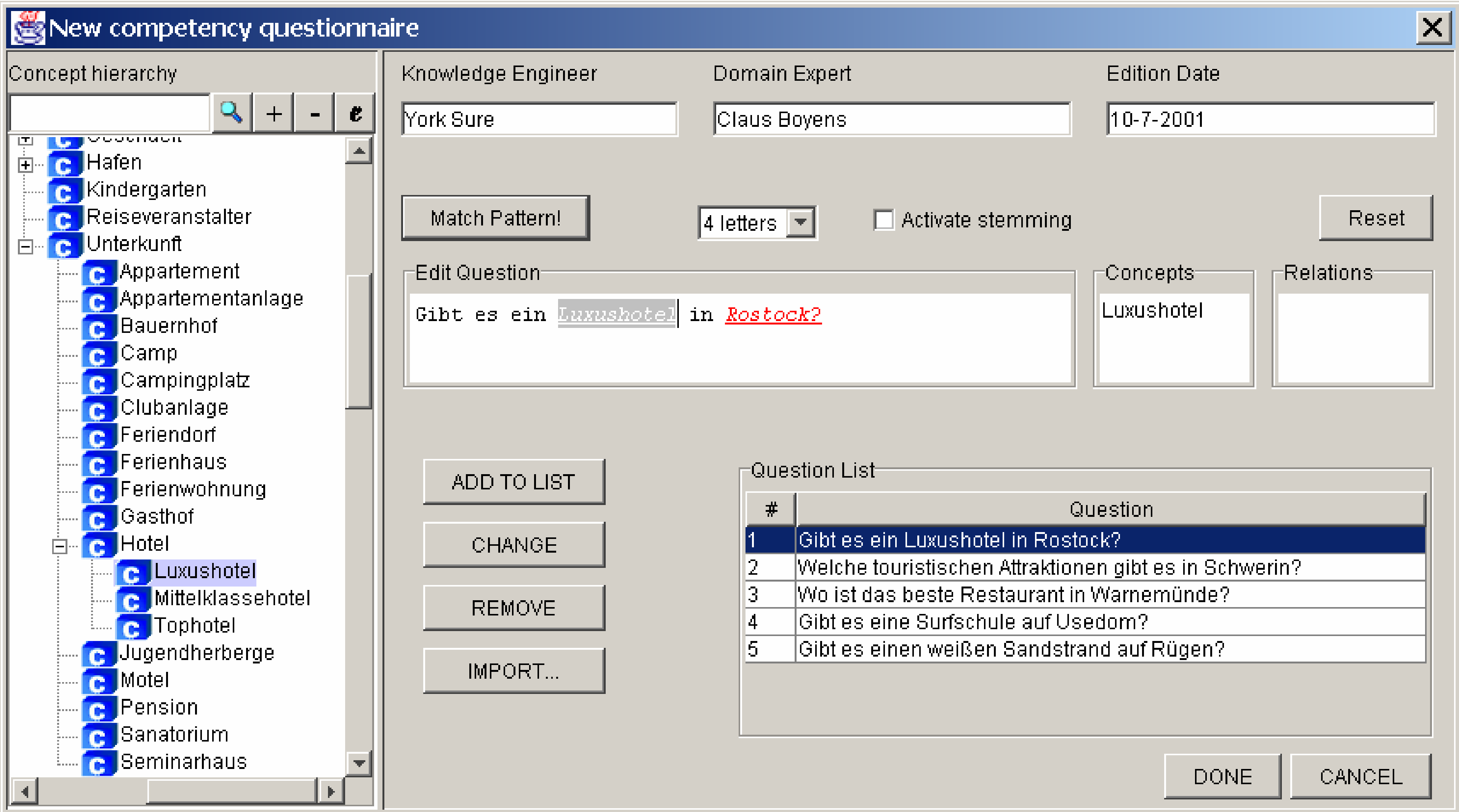
Kick-off

Competency questions



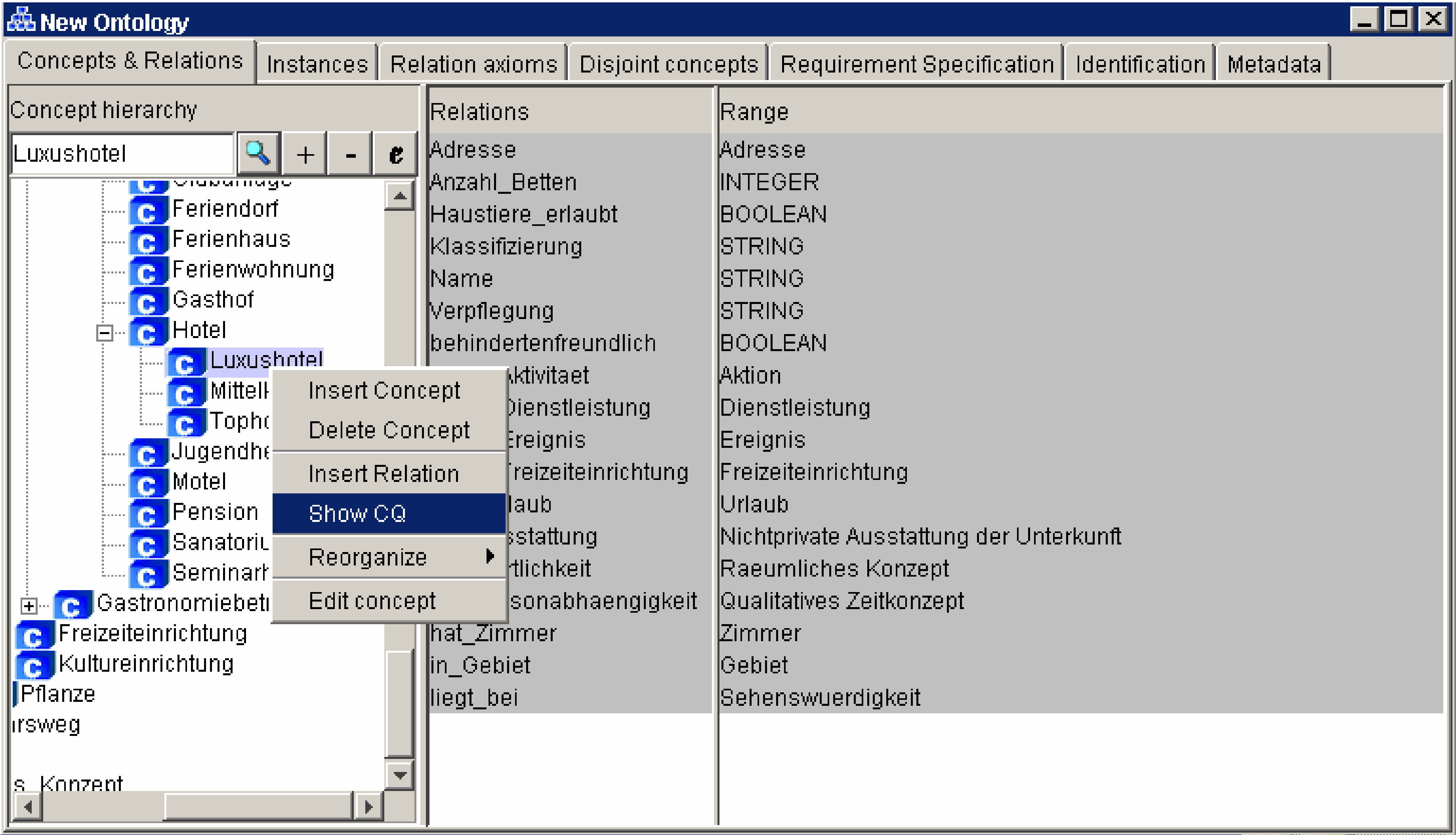
Kick-off

Competency questions



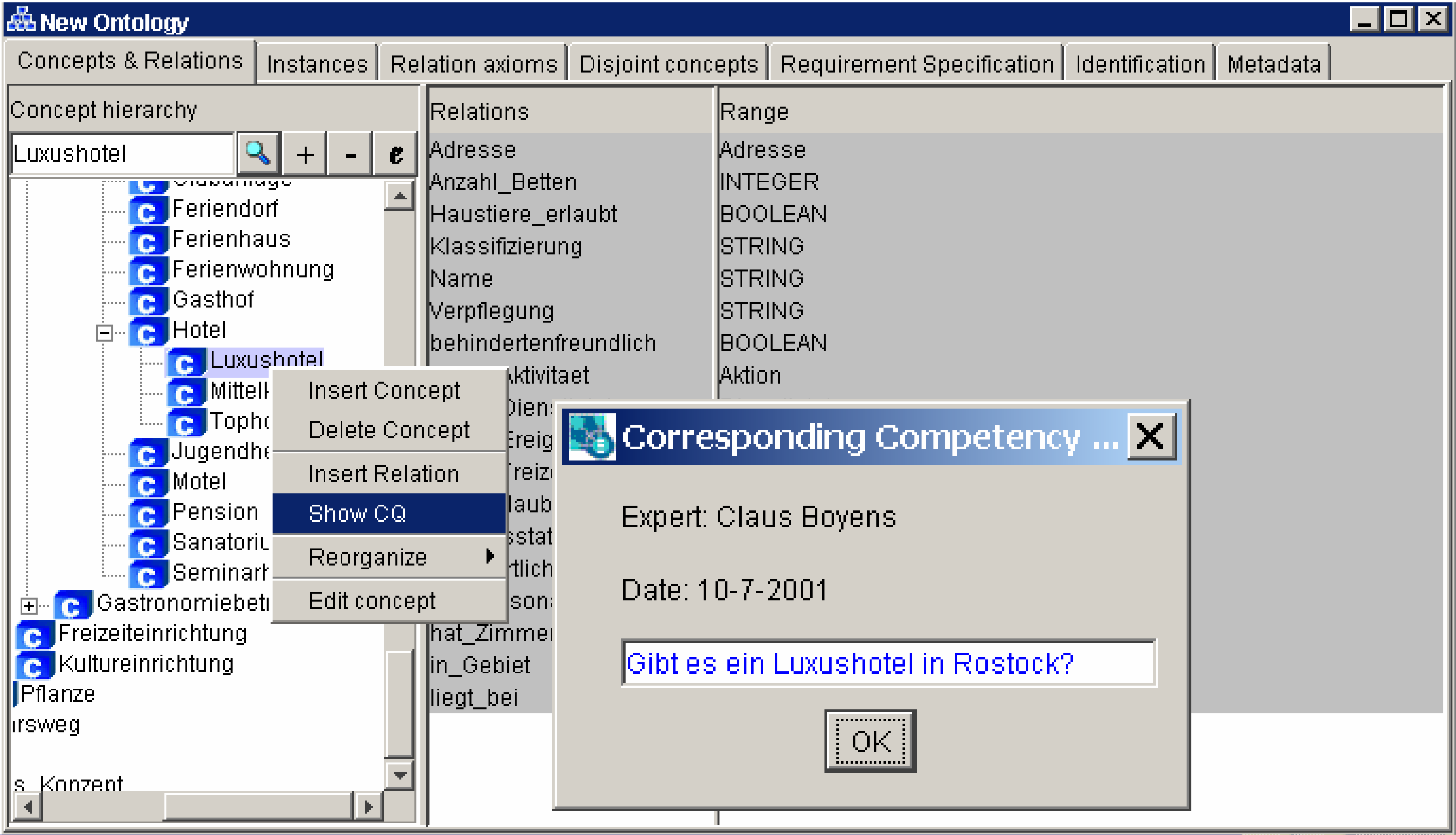
Kick-off

Traceability



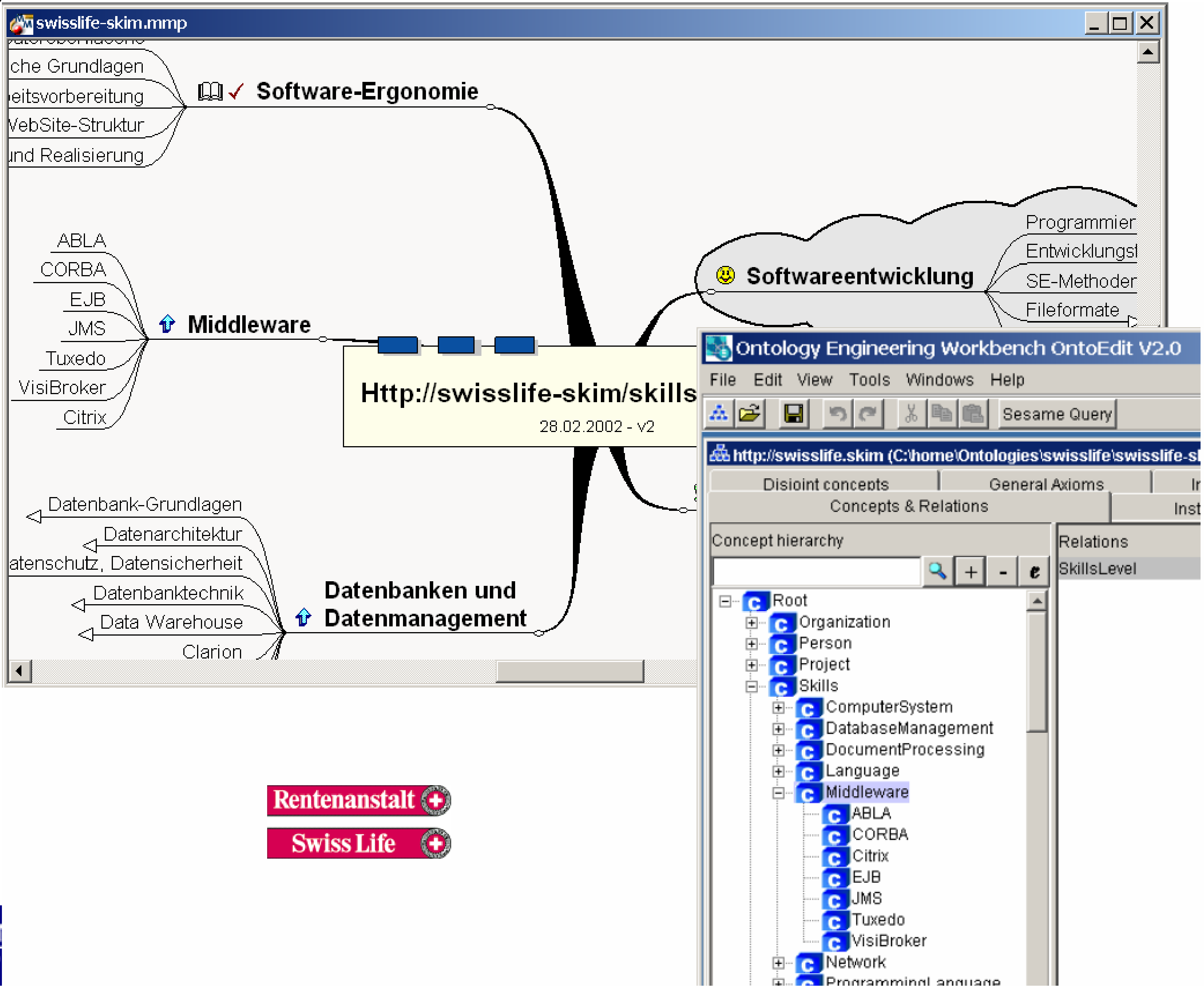
Kick-off

Traceability



Kick-off

Brainstorming, Structuring, Formalisation



Rentenanstalt

Swiss Life

Kicko ff

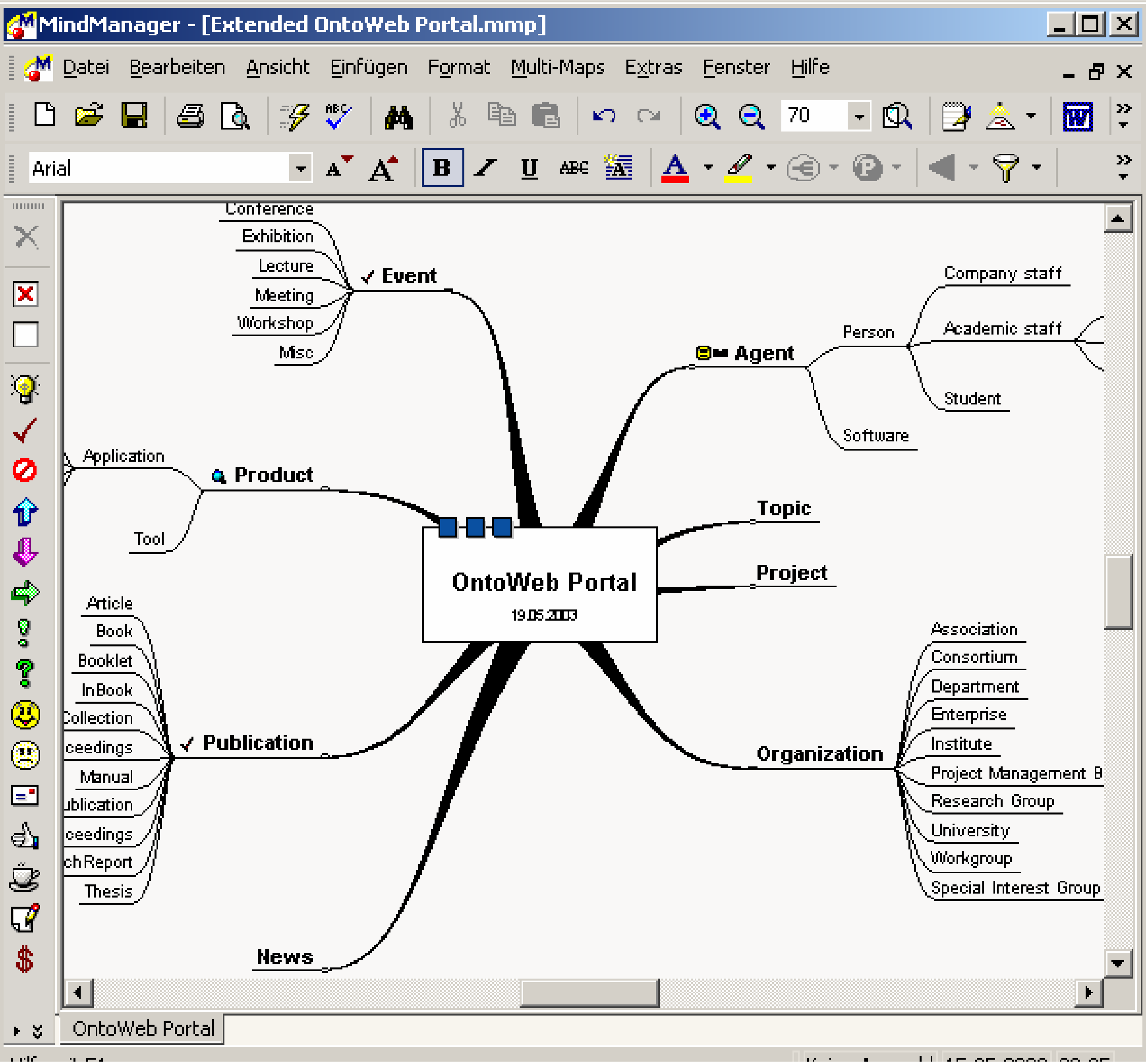
Mind2Onto

- **Task:** Collaborative capturing of domain knowledge through domain experts and modelling experts

- **Problem**: Collaboration with domain experts who have:
  - **No experience** with modelling
  - **No time** for modelling

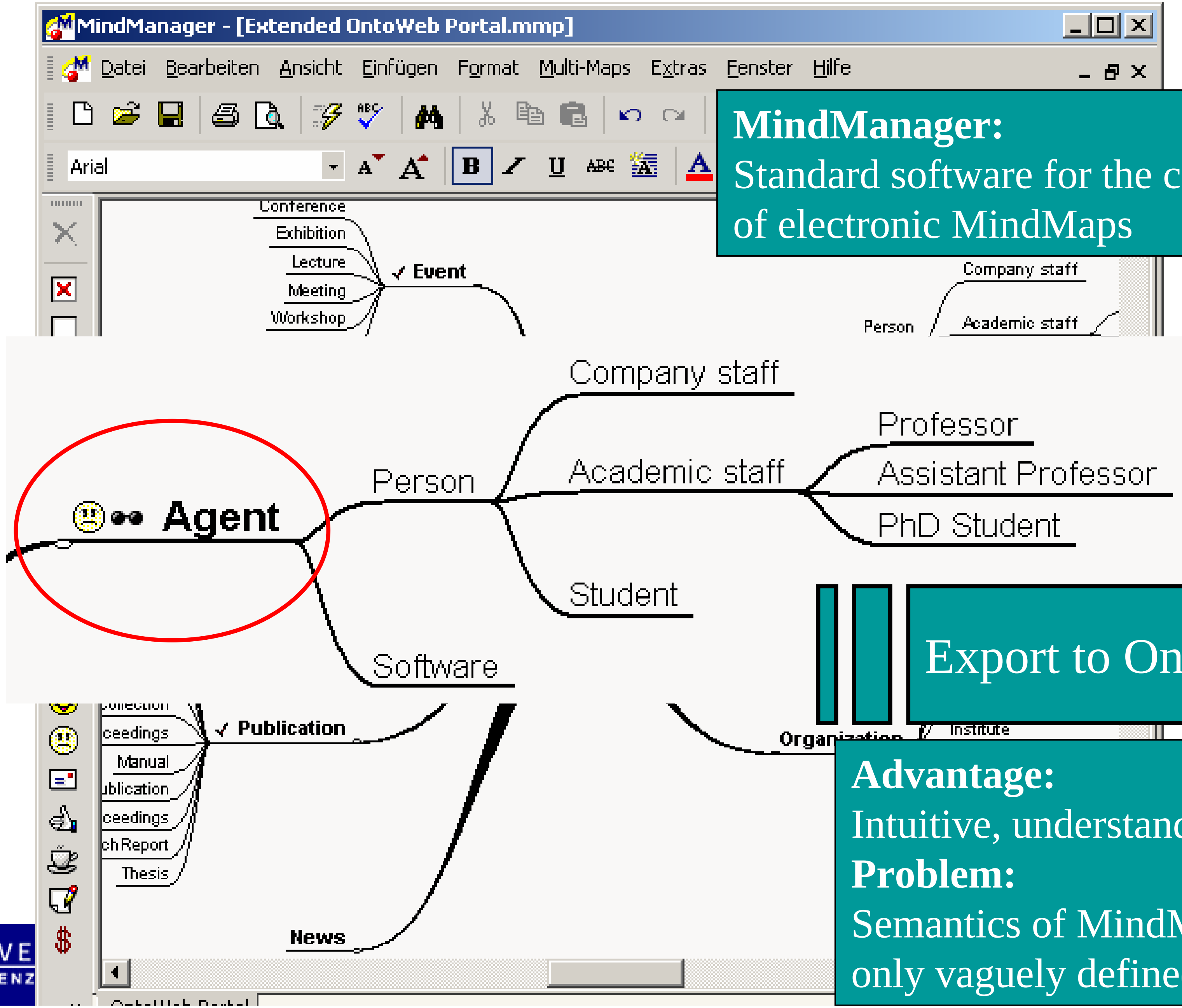
Kickoff

Mind2Onto



Kick off

Mind2Onto



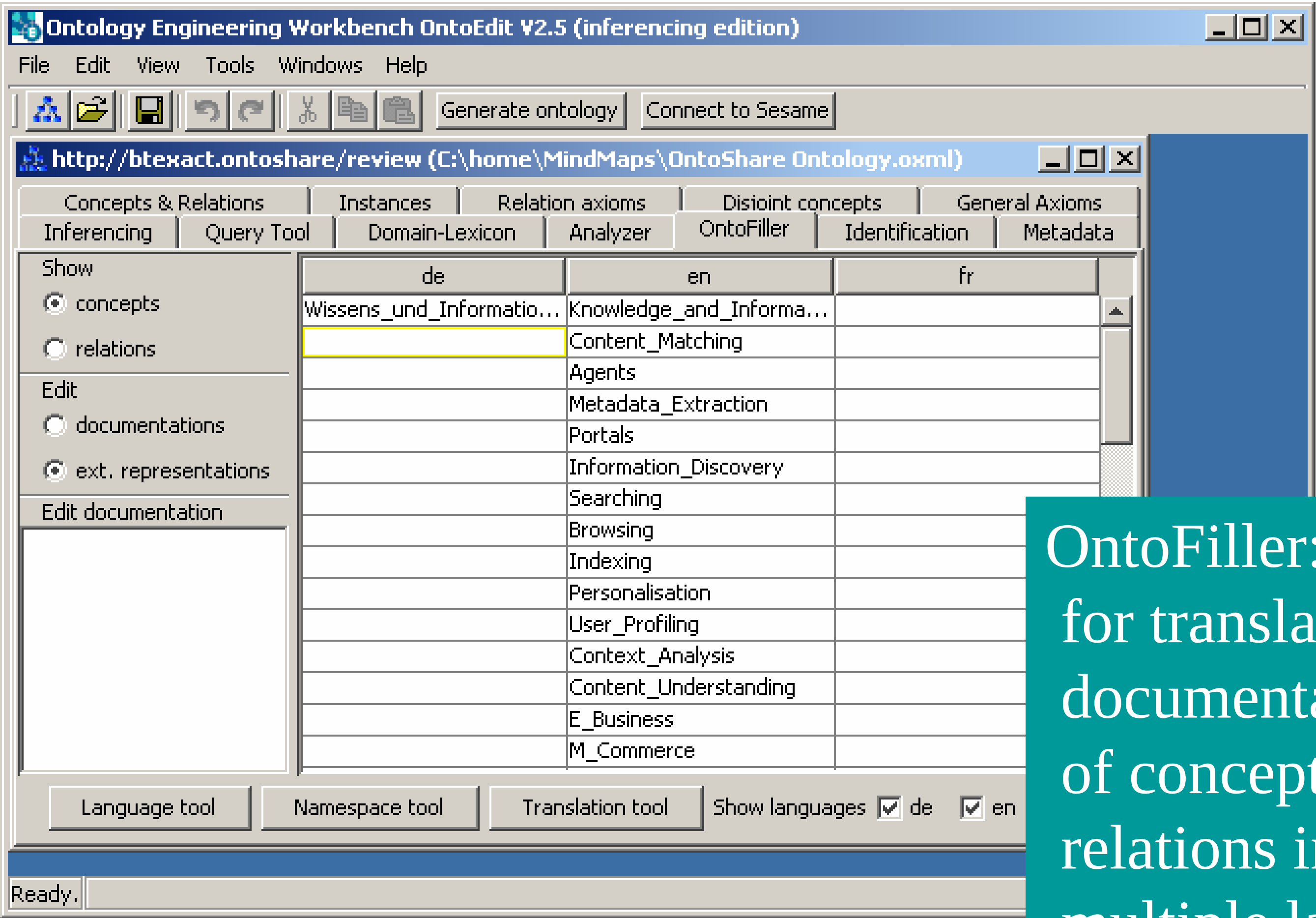
MindManager:
Standard software for the creation of electronic MindMaps

Export to OntoEdit

Advantage:
Intuitive, understandable
Problem:
Semantics of MindMaps only vaguely defined

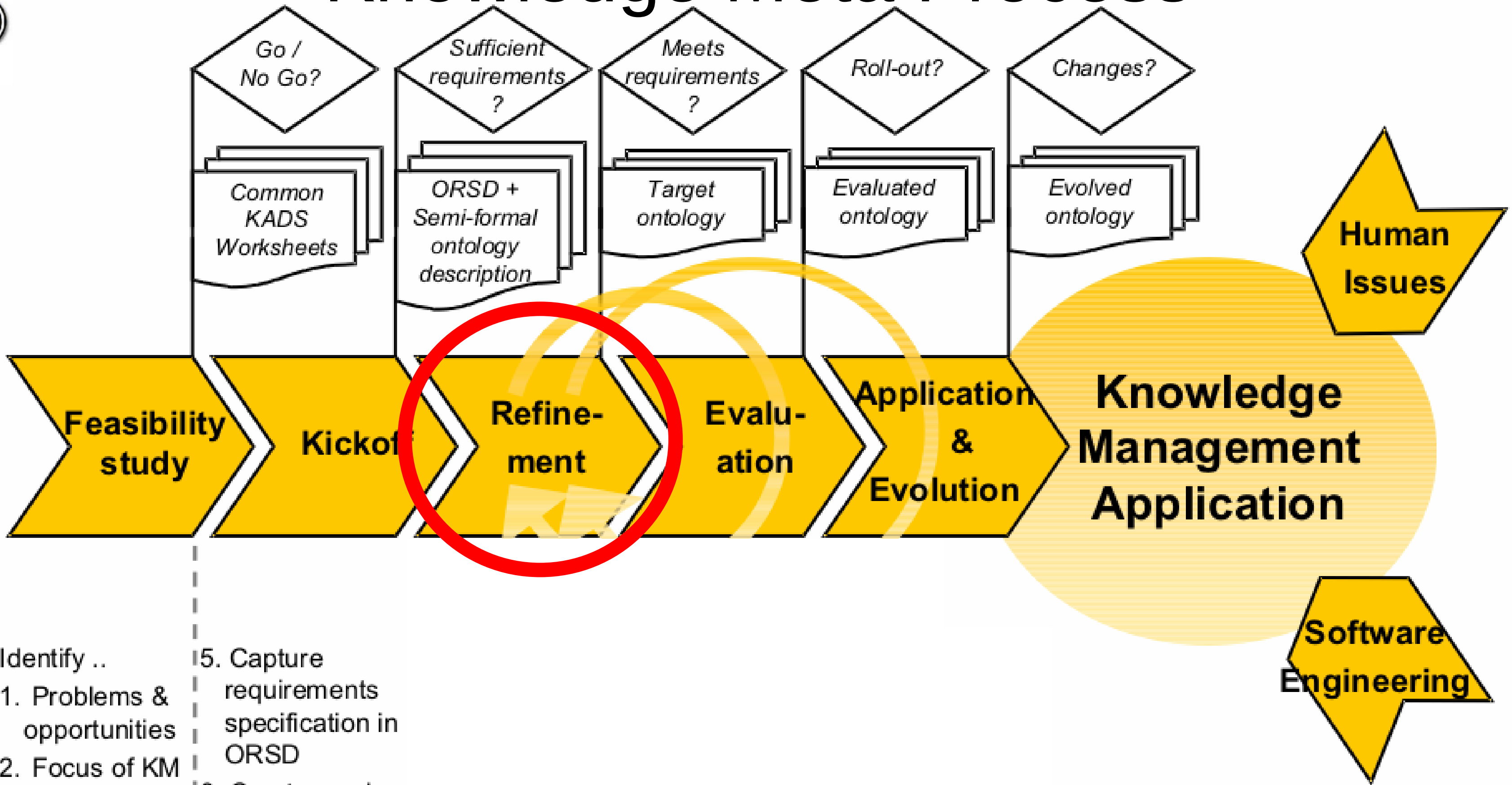
UNIVE KOBLENZ

OntoEdit/OntoFiller



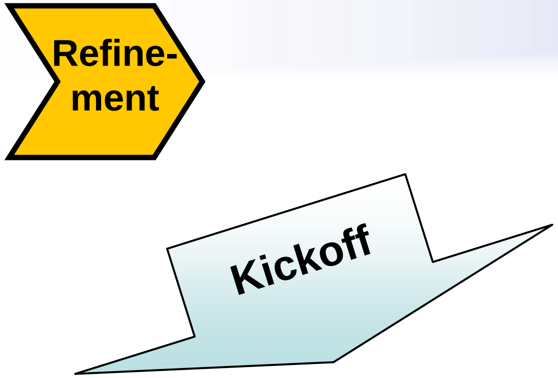
OntoFiller: Support for translation and documentation of concepts and relations in multiple languages

OTK Methodology:
Knowledge Meta Process



Identify ..
1. Problems & opportunities
2. Focus of KM application
3. (OTK-) Tools
4. People

5. Capture requirements specification in ORSD
6. Create semi-formal ontology description

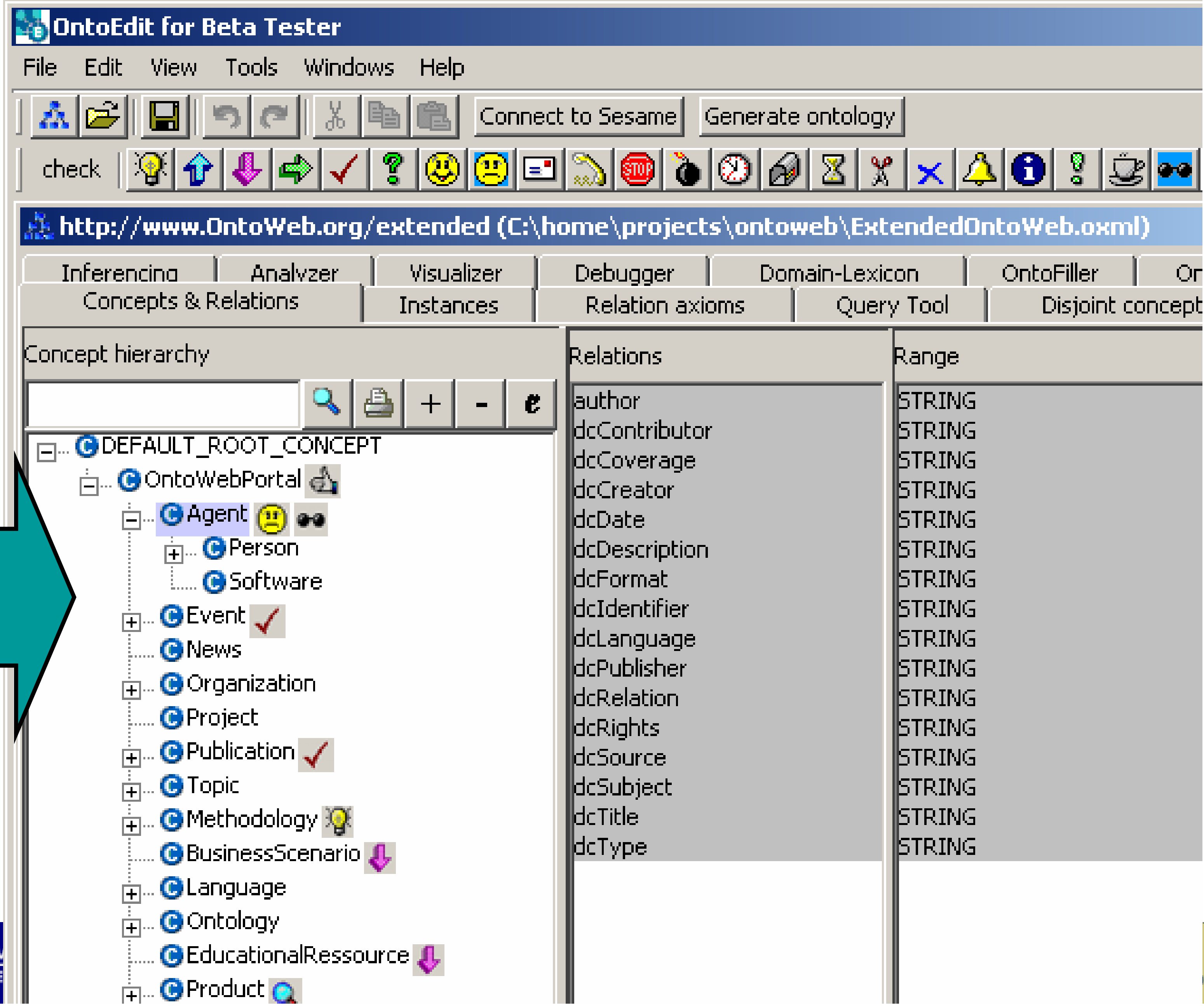


Refinement

- Knowledge elicitation with domain experts
  - Refine concepts and relations
  - Typically axioms are identified
- Formalize
  - E.g. F-Logic, DAML+OIL
  - Axioms depend on language capabilities
- Develop and refine ***ontology***

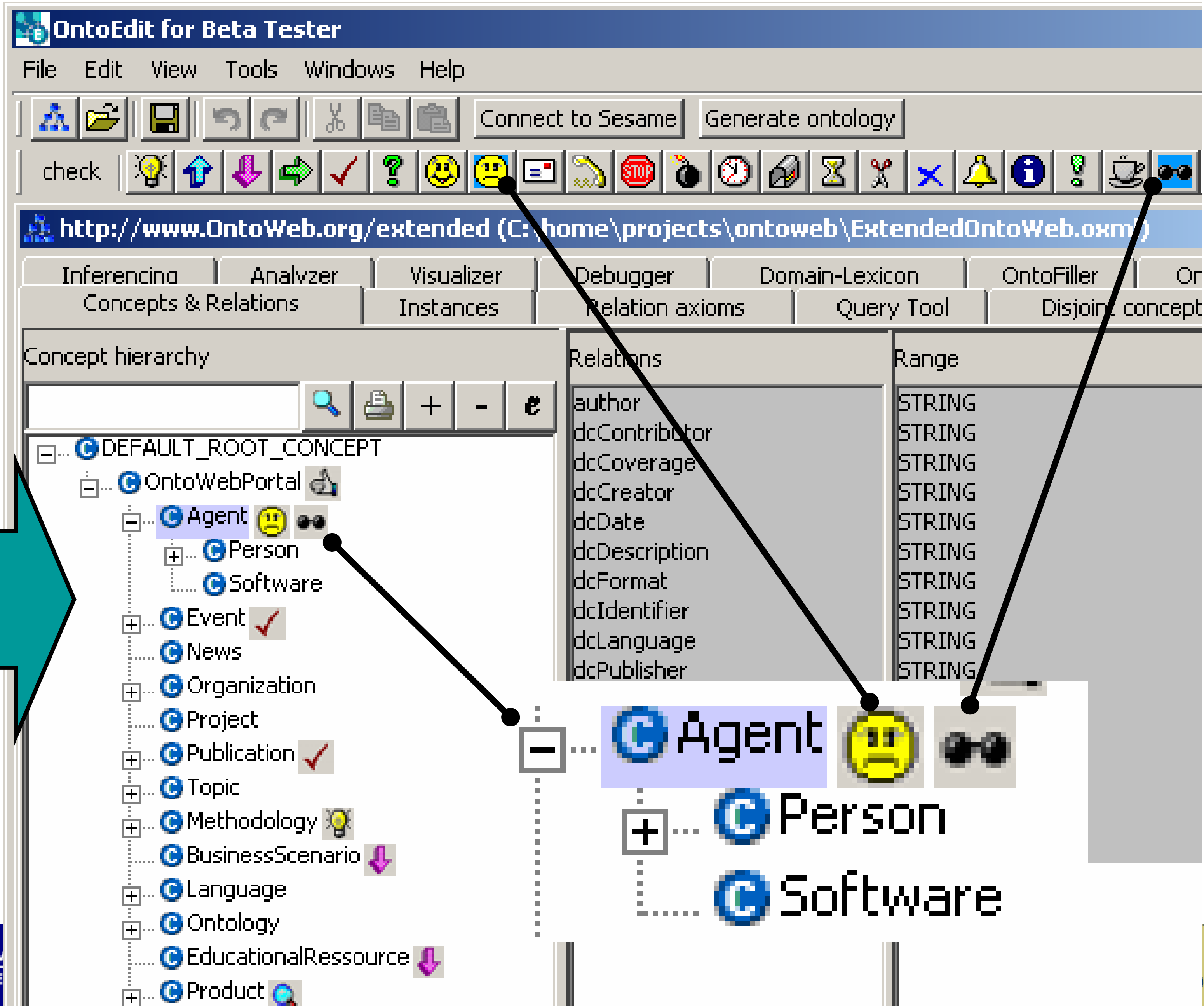
Refine-ment

Mind2Onto



Refine-
ment

Mind2Onto



Refine-ment

Inferencing

Theoretical Issues

- F-Logic
  - Object-oriented
  - Deductive Database-oriented
  - Well-founded semantics

Practical Issues

- Namespace mechanism: Ontologies/Ontology Parts -> modules
- Switch-off definitions:
  - For testing
  - For fast executions without consistency checks
- DB Connectors: map DB tables via JDBC
- User-definable built-Ins
- Extensive API:
  - remotely connect to the inference engine
  - import and export several standards (e.g., RDF(S))

Refine-
ment

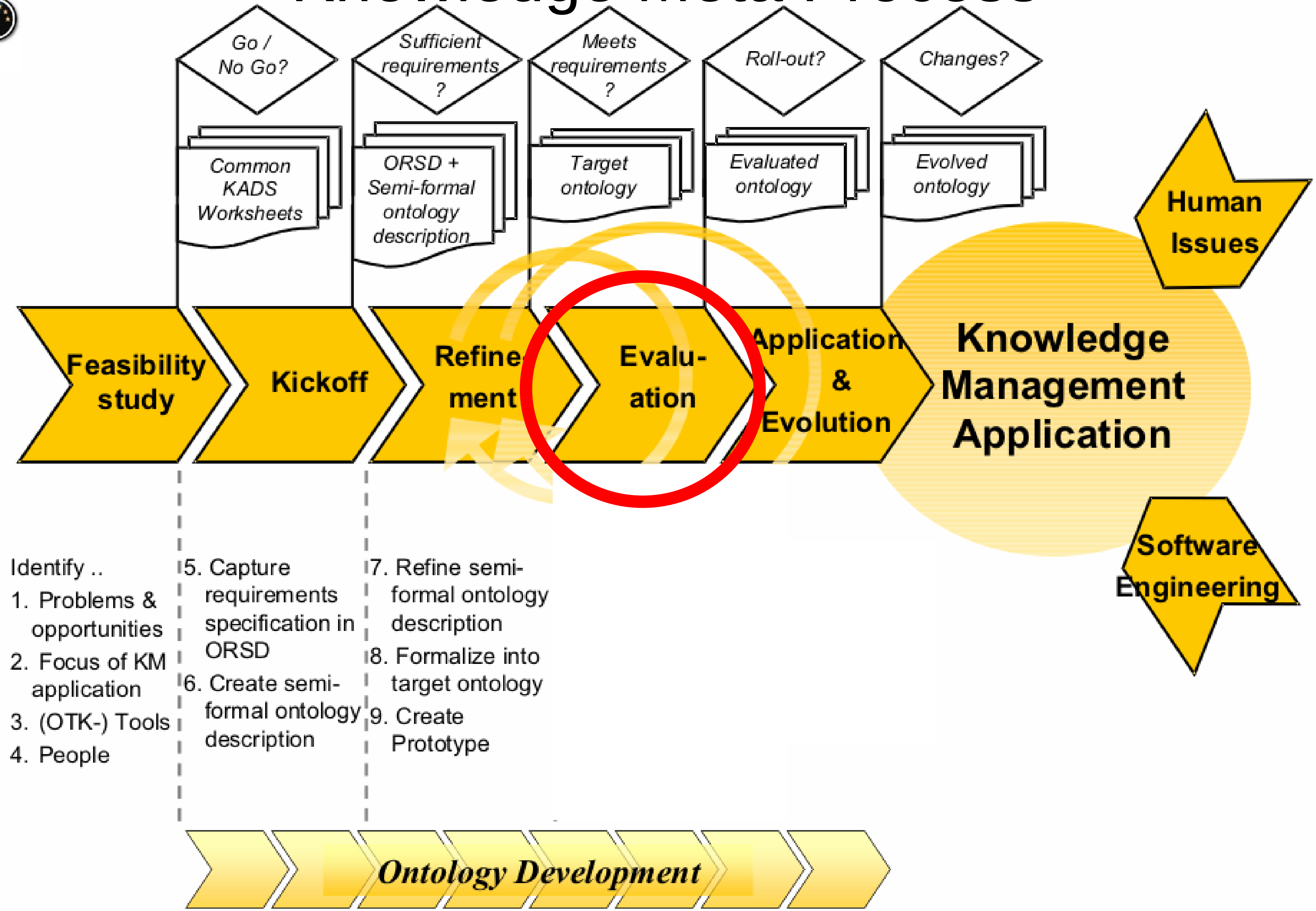
Exploit Inferencing

- Hook in existing resources with inferencing
  - Jdbc
  - Rules
- Construct axiom libraries
  - Temporal reasoning
  - PartWhole reasoning
  - ...

- Selective axiom applications
  - F-Logic semantics: E.g. type coercion at concept level
  - Domain specific consistency: non-cyclic hasPart
  - Axioms for modeling policies
  - Debugging

Contrast: OilEd

OTK Methodology:
Knowledge Meta Process



Refi
ne-
men
t

Evaluation

- Check requirements (ORSD)
  - Are all CQs answered?
  - Is the ontology within the scope?

- Test in target application
  - Analyze usage pattern
- Deploy application(s)

Eva-
luation

OntoClean

- **Task:** Formal evaluation of ontologies

- Well-known methodology:

  OntoClean [Welty & Guarino, 2001]
  - Aims at **„cleaning"** of hierarchies
  - Based on philosophical notions
    - „essence", „rigidity", „identity", „unity" ... etc.

- **Implementations:** For F-Logic & OWL

Eva-luation

OntoClean: Definitions

„Essence“: A property is essential for an individual *iff.* it necessarily holds for that individual.

Example: York is *necessarily* a person.

„Rigidity“

- A property is „rigid“ (+R) *iff.* it is **necessarily essential for all** its individuals.
- A property is „non-rigid“ (-R) *iff.* it is **not essential for some** of its individuals.
- A property is „anti-rigid“ (~R) *iff.* it is **not essential for all** its individuals.

Example: „Person“ is necessarily an essential property for all its individuals.

- There exist similar definitions for „identity“ (+I, -I, +O, -O), „unity“ (+U, -U, ~U), „dependency“ (+D, -D), ... etc. ...

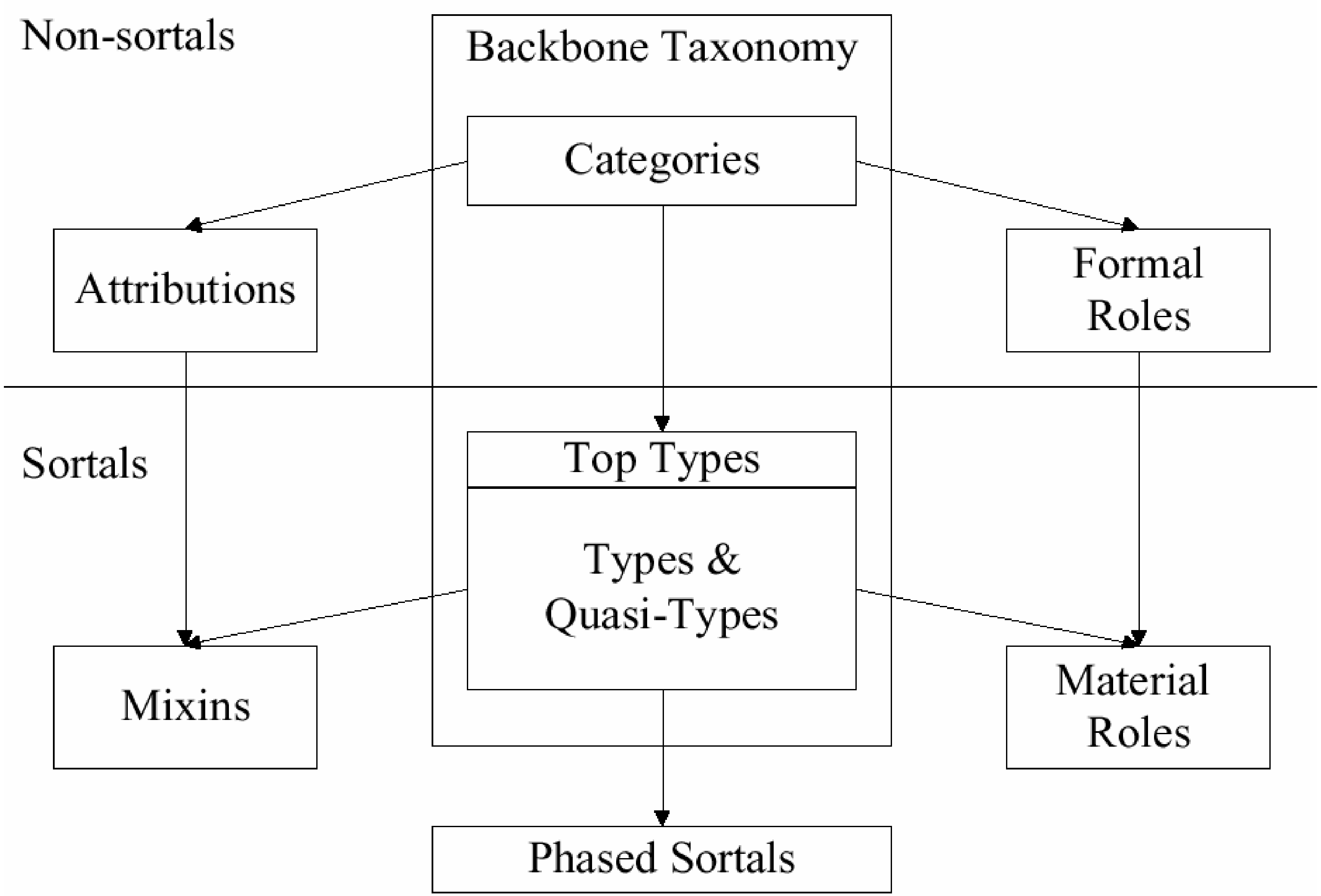
Eva-
luation

OntoClean: Classification & ideal structure

+O	+I	+R	+D	Type	Sortal
			-D		
-O	+I	+R	+D	Quasi-type	
			-D		
-O	+I	~R	+D	Material role	
-O	+I	~R	-D	Phased sortal	
-O	+I	¬R	+D	Mixin	
			-D		
-O	-I	+R	+D	Category	Non-sortal
			-D		
-O	-I	~R	+D	Formal Role	
-O	-I	~R	-D	Attribution	
		¬R	+D		
			-D		
+O	-I			incoherent	
	+I	~R			
		-R			

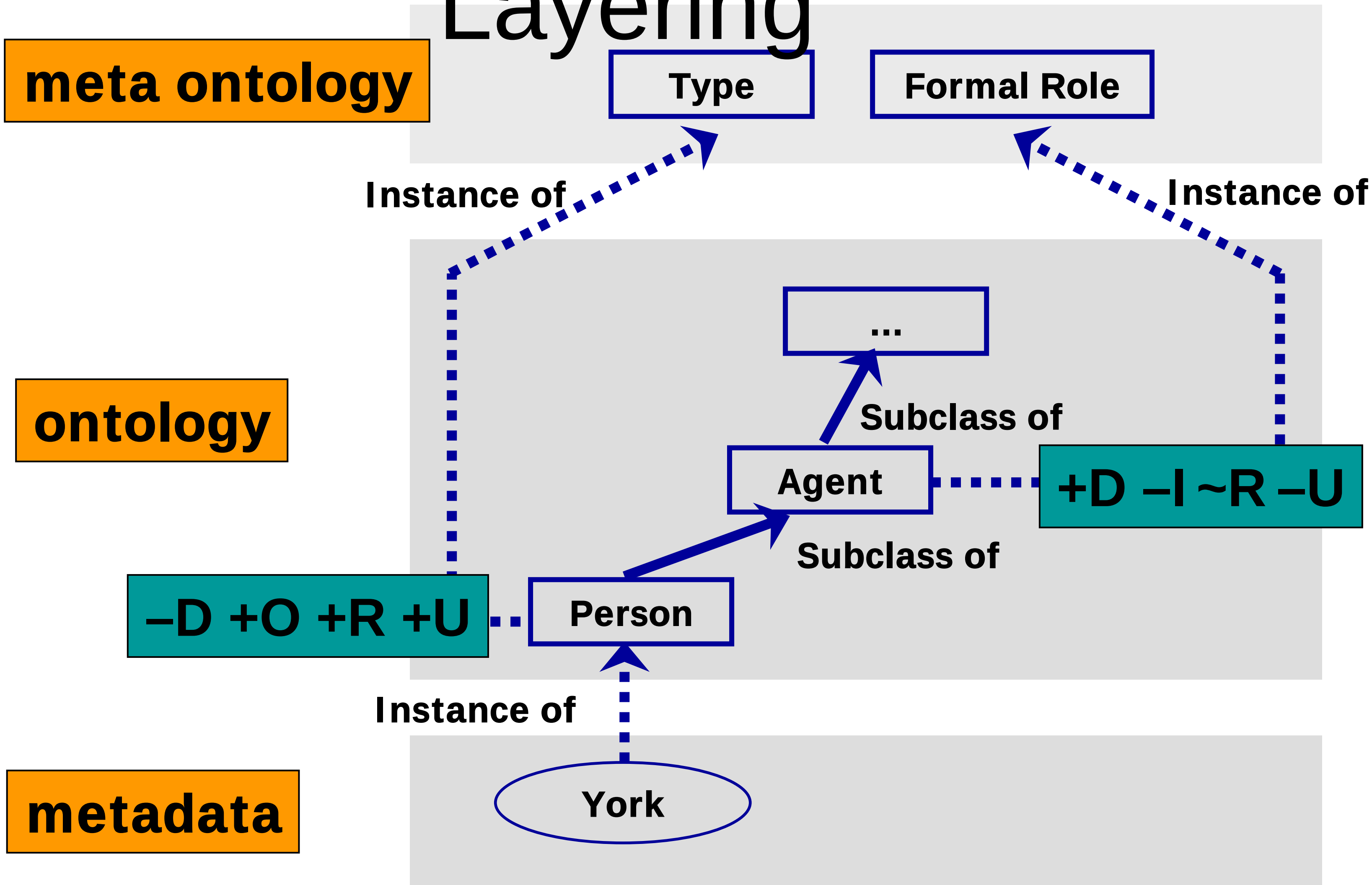
See:
[Welty & Guarino, 2001]

OntoClean: Classification & ideal structure



See: [Welty & Guarino, 2001]

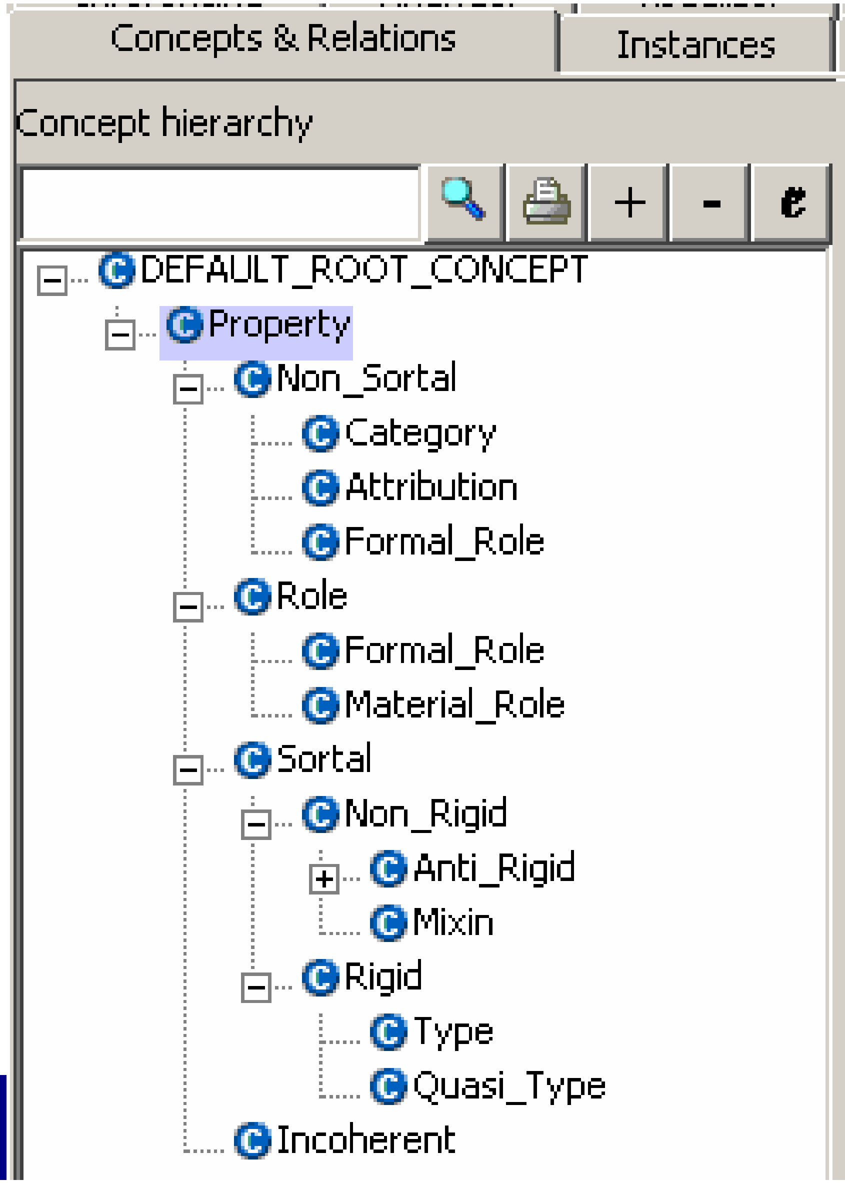
OntoClean: Layering



Eva-luation

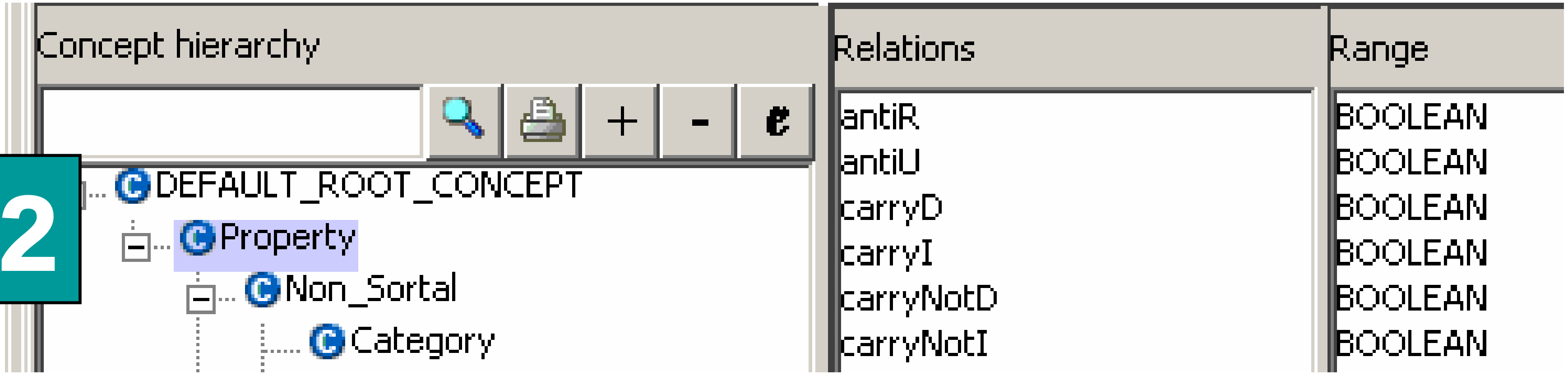
OntoCleanPlugin: Formalisation of meta ontology

1



Eva-luation

OntoCleanPlugin: Formalisation of meta ontology

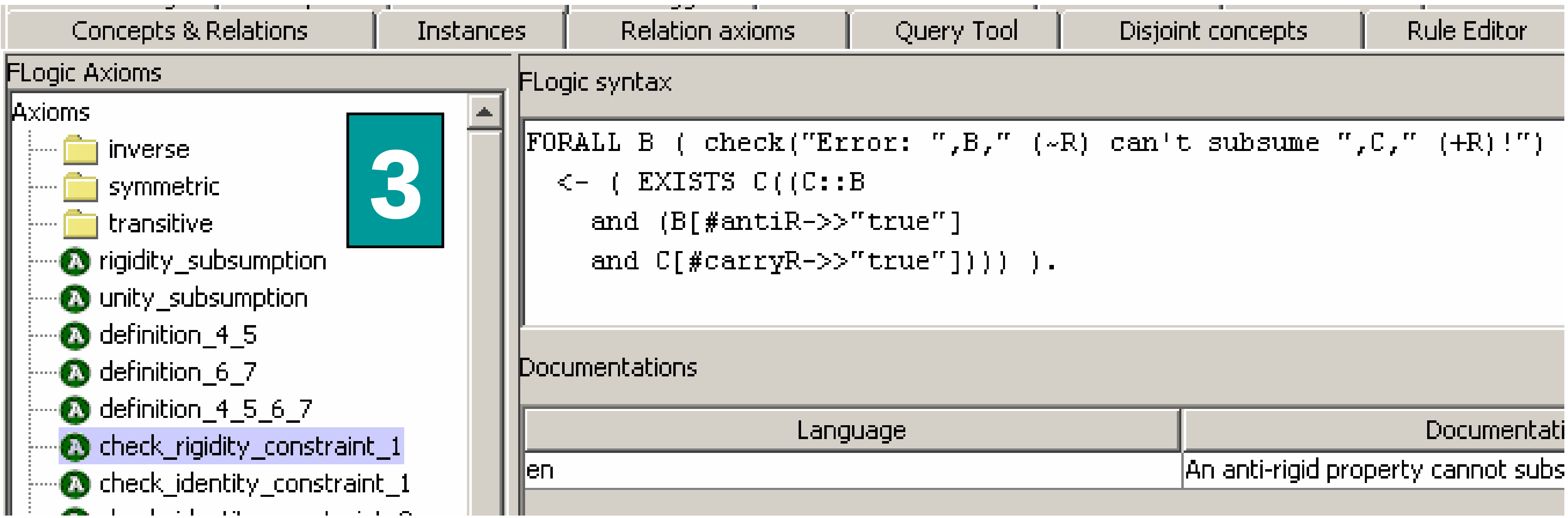


2

Uppermost concept „Property“ of the ***meta ontology*** has attached all relations necessary for classifying concepts of an ***ontology***

Eva-
luation

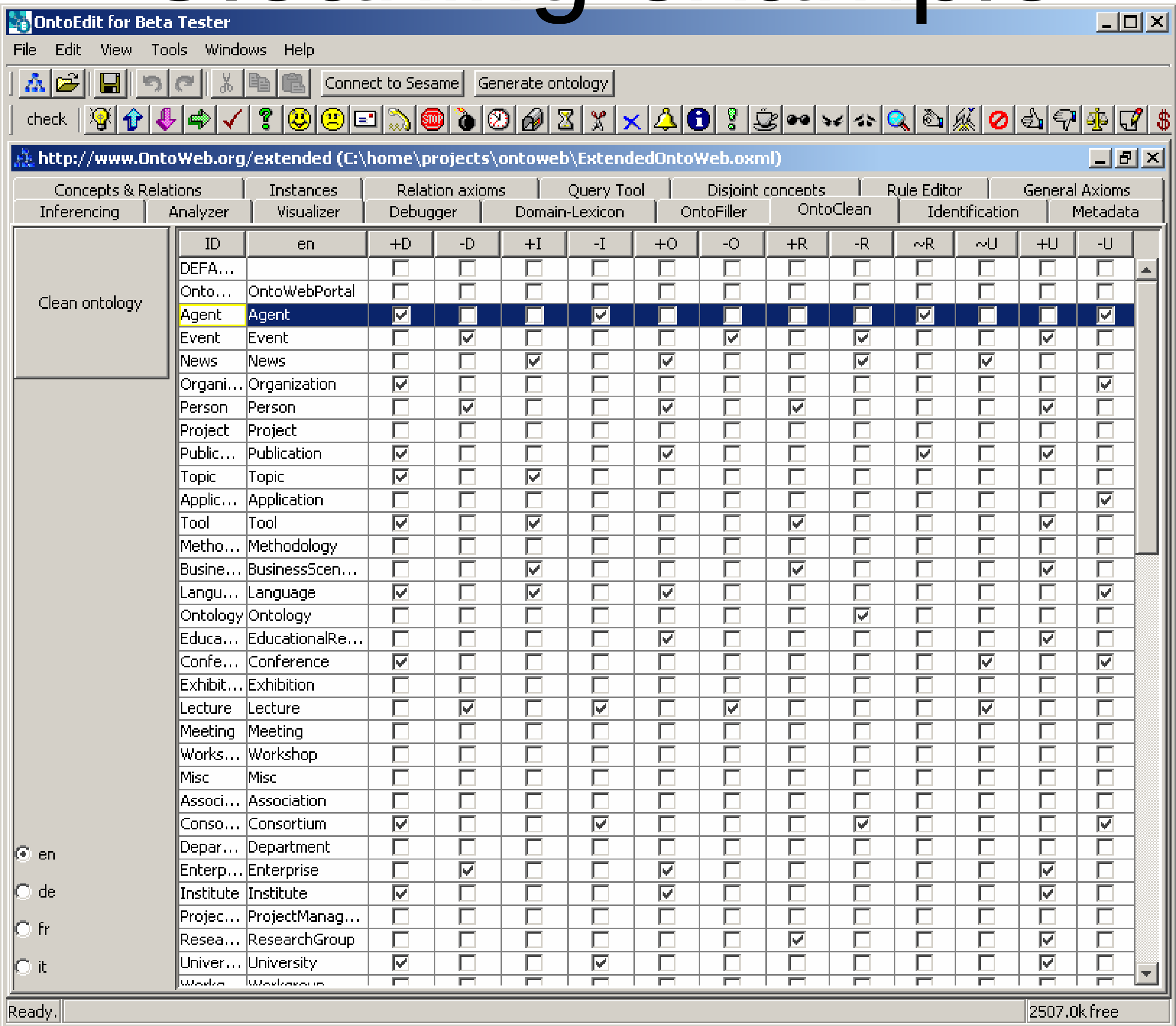
OntoCleanPlugin: Formalisation of axioms



- Anti-rigid concepts (~R) cannot have rigid subconcepts (+R)
- *Etc.*

OntoCleanPlugin: Cleaning example

Eva-
luation

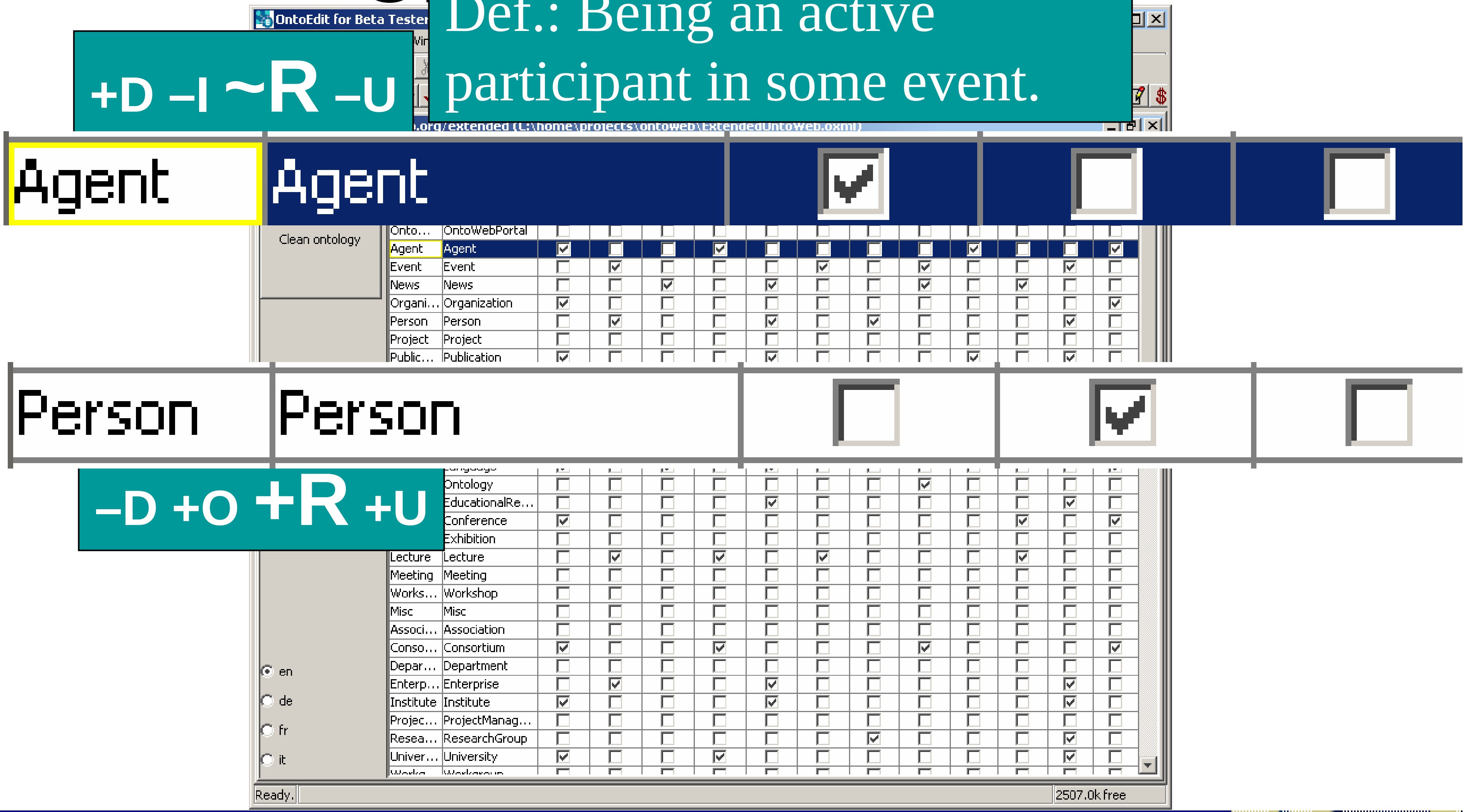


Eva-
luation

OntoCleanPlugin: Cleaning example

Def.: Being an active participant in some event.

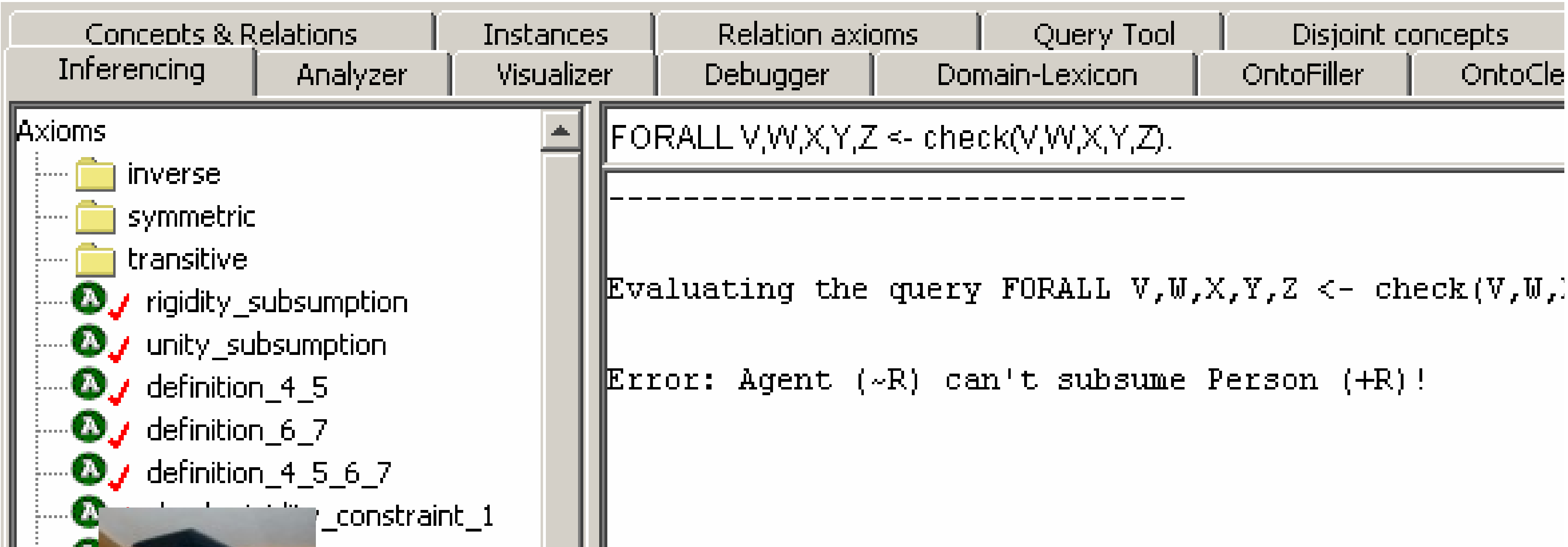
+D –I ~R –U

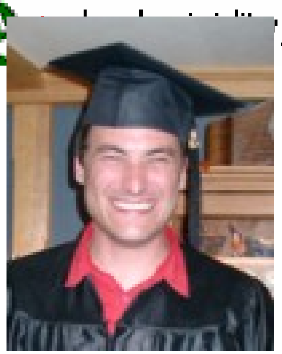


–D +O +R +U

Eva-luation

OntoCleanPlugin: Cleaning example

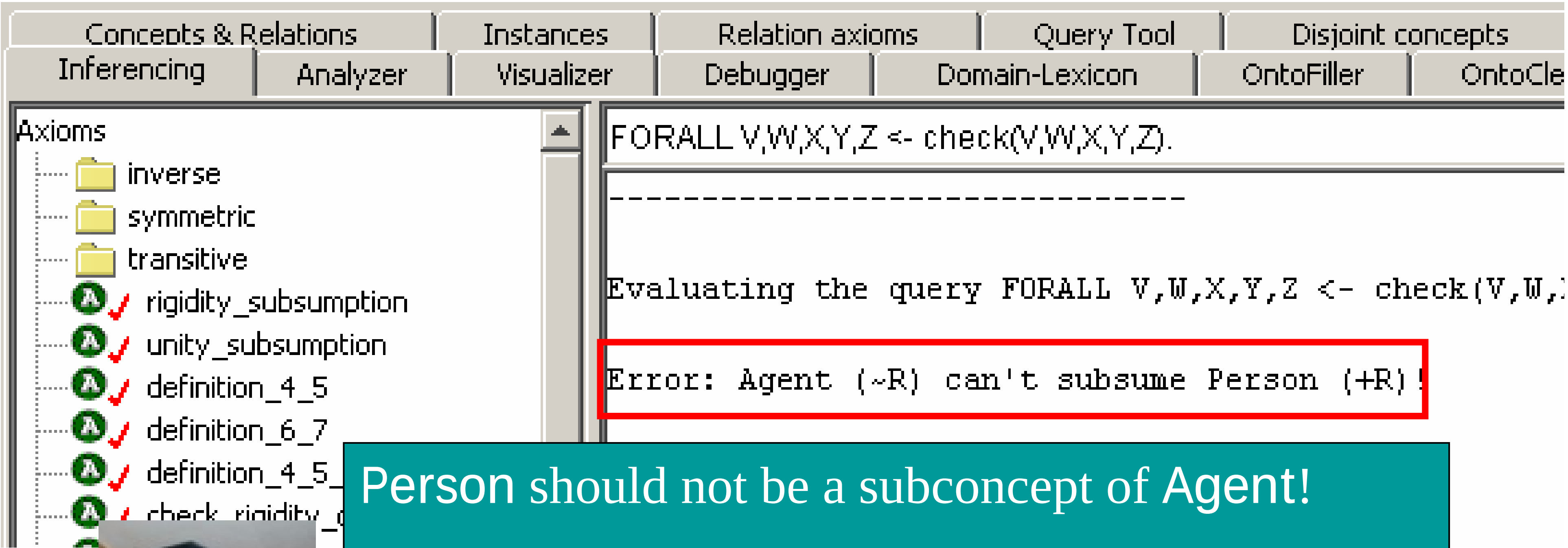




„Is York an agent?“

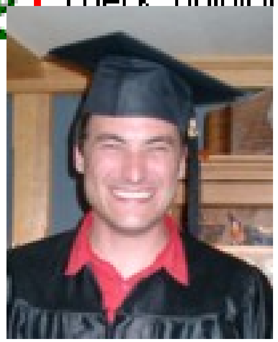
Eva-luation

OntoCleanPlugin: Cleaning example



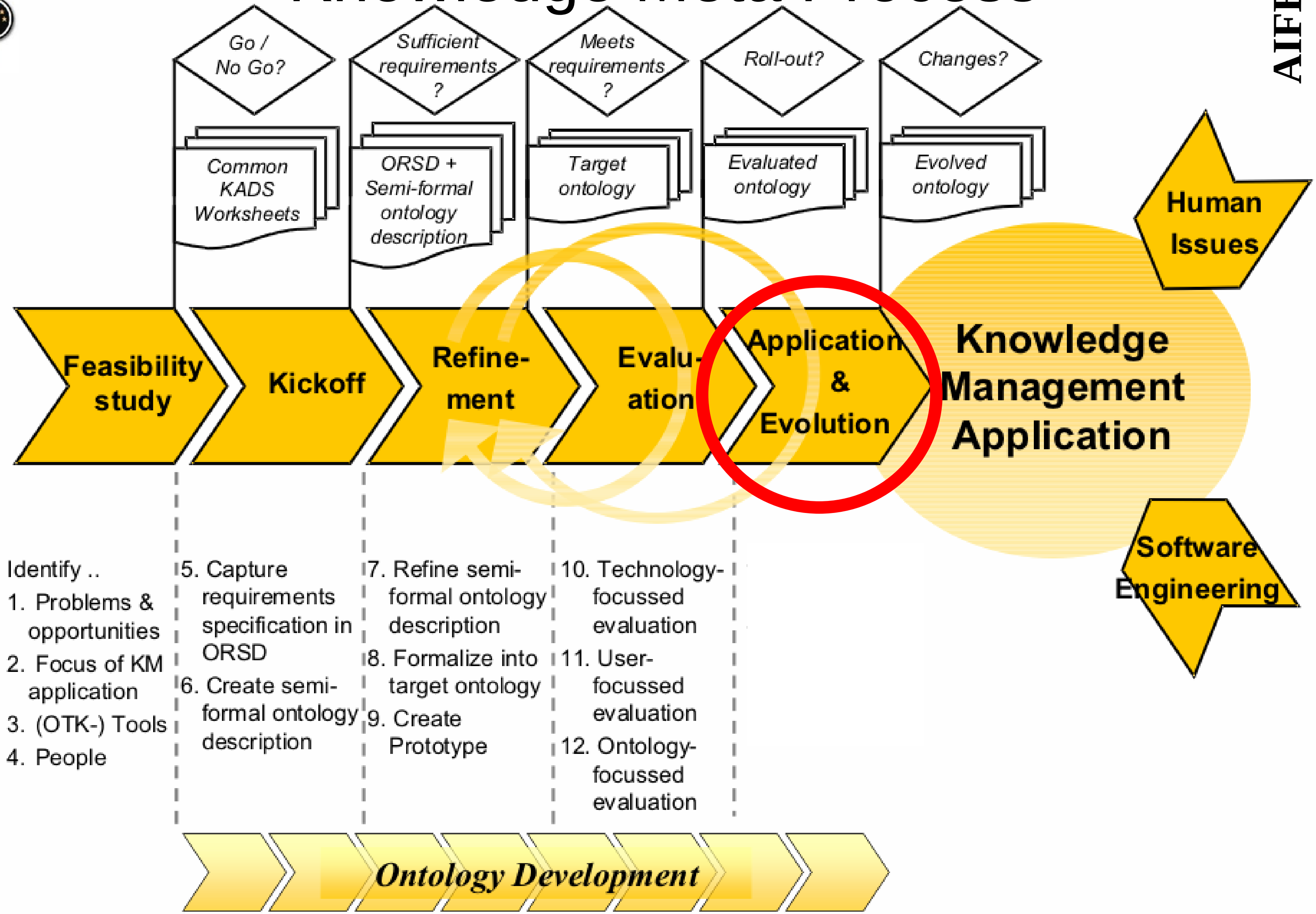
Person should not be a subconcept of Agent!

Interpretation: Persons ***can be*** agents, but persons are not necessarily agents.



„Is York an agent?“

OTK Methodology:
Knowledge Meta Process



Evo-
lution

Worksheet for life cycle aspects of ontology

- Who is going to maintain it?
- Who is going to pay for it?
- What is the resulting quality (increase, decrease)?
- How large are the network costs (cost of negotiation grows quadratic with number of participants)?
- What is the expected life time of the ontology?
- How brittle is it with regard to updates?
- What error types will occur/are relevant?

Evo-
lution

Worksheet for life cycle aspects of metadata

- ala ontology

Rule of thumb – costs:

•Hardware 1

•Software 10

•Daten 100

- Co-ordinated change of data and metadata?
- Co-ordinated change of ontology and metadata?

- Cold start (chicken-and-egg) problem: A problem? How to overcome?
- Granularity of metadata envisaged: classification, identification of people/events/relationships/etc.

Evo-
lutio
n

Coordination of metadata & ontology

- Match or mismatch between the two,
  - E.g. classification only, but ontology about transitive relationships

Evo-
lution

Type-1 Error

- False Positive
  - Often dominating problem in company internal IR
  - It can be more costly to learn about all low-price provider of pens than to just select from a sample

Evo-
lution

Type-2 Error

- False negative: Positive example not detected as such
  - Often not critical for information retrieval
    - „show me bookstores who sell the `CommonKADS‘ book“
  - Often critical for B2B operations
    - „whether `6000 computer‘ is mapped to
      ‚IBM RS/6000 SP system‘ or to
      `HP OmniBook Laptop 6000‘ is a large difference with regard to price and performance“

Evo-
lution

Refined Error types (Halo Project)

- 1. (MOD) Knowledge Modeling: the ability of the knowledge engineer to model information/write axioms
- 2. (IMP) Knowledge Implementation/Modeling Language: the ability of the representation language to accurately represent axioms
- 3. (INF) Inference and Reasoning: the ability of the inference engine to “find the needle in the haystack”
- 4. (KFL) Knowledge Formation and Learning: the ability of the system (KB + inference engine) to acquire and merge knowledge through automated and semi-automated techniques
- 5. (SCL) Scalability: the ability of the KB to scale

  – http://www.haloproject.com

Evo-
lution

Refined Error types II (Halo Project)

- 6. (MGT) Knowledge Management: the ability of the system to maintain, track changes, test, organize, document; the ability of the knowledge engineer to search for knowledge
- 7. (QMN) Query Management: the ability of the system to robustly answer queries
- 8. (ANJ) Answer Justification: the ability of the system to provide justifications for answers in the correct context and resolution
- 9. (QMT) Quality Metrics: the ability of the developers to determine how “good” the knowledge base is at any given point in its evolution
- 10. (MTA) Meta Capabilities: the system's ability to utilize meta-reasoning or meta-knowledge

Evo-lution Ontology Evolution: Technical aspects

- **Ontology development is necessarily an iterative and a dynamic process**
- **Ontologies must be able to evolve for a number of reasons:**
  - **Application domains and user‘s needs are changing**
  - **System can be improved**
- **Developing ontologies is expensive, but evolving them is even more expensive**

Evo-
lution

Requirements for ontology evolution

Basic requirement

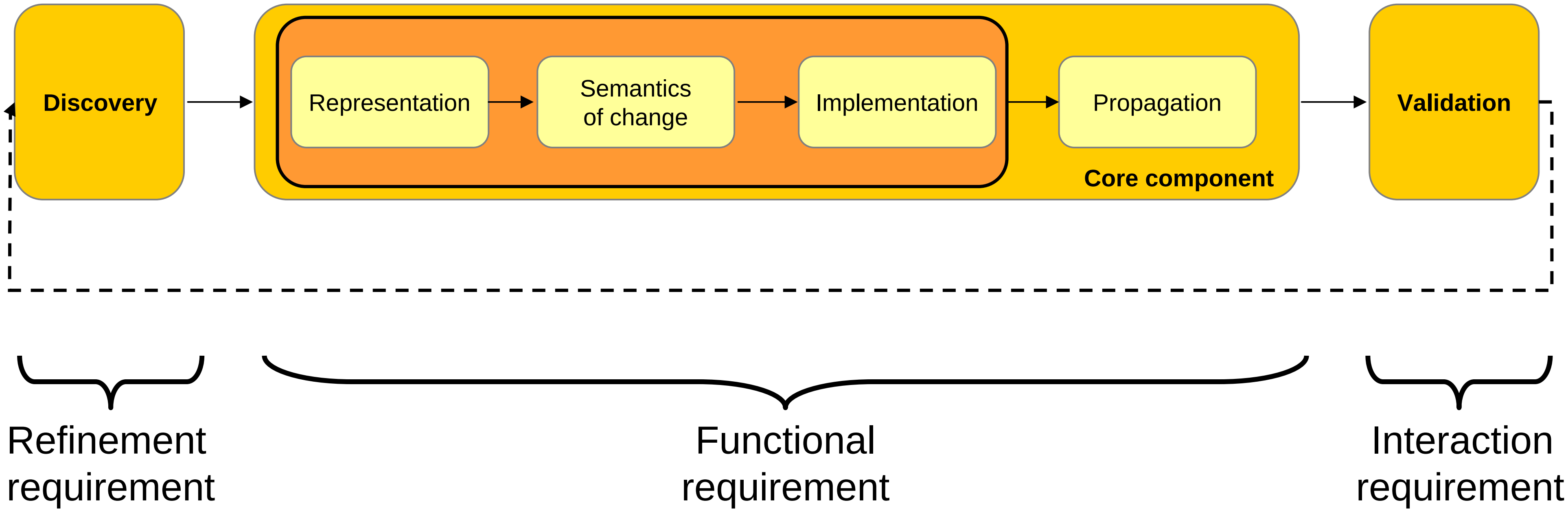
- Functional requirement:
  - enable the handling of the required changes
  - ensure the consistency of the underlying ontology and all dependent artifacts

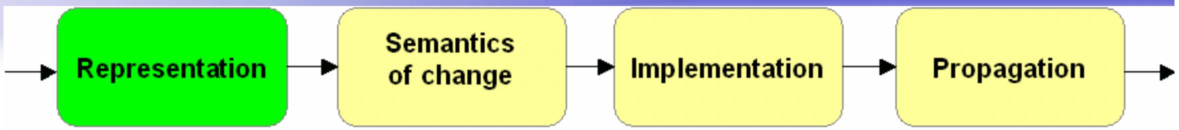
Extended requirements

- Interaction requirement – supports the user to manage changes more easily
- Refinement requirement –offers advice to the user for continual system refinement

Evo-
lution

Ontology Evolution Process





Ontology Evolution – Change representation

- Elementary changes
  - ➢ They can not be decomposed into simpler ones
  - ➢ They heavily depend on the underlying ontology model

MoveConcept ≠ (RemoveSubConcept + AddSubConcept)

- Composite changes
  - ➢ They are more powerful
  - ➢ They have coarser granularity
  - ➢ They have often more meaningful semantics

Ontology Evolution – Change representation

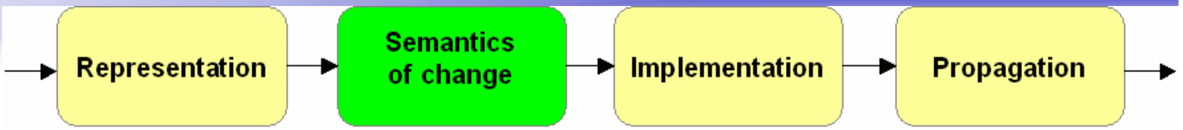
Composite change	Description
Move concept	Move a concept from one parent to another.
Merge concepts	Replace several concepts with one and aggregate all instances.
Extract subconcepts	Split a concept into several subconcepts and distribute properties among them.
Extract superconcept	Create a common superconcept for a set of unrelated concepts and transfer common properties to it.
Extract related concept	Extract related information into a new concept and relate it to the original concept.
Shallow concept copy	Duplicate a concept with all its properties.
Deep concept copy	Recursively apply shallow copy to all subconcepts of a concept.
Pull up properties	Move properties from a subconcept to a superconcept.
Pull down properties	Move properties from a superconcept to a subconcept.
Move properties	Move properties from one concept to another concept.
Shallow property copy	Duplicate a property with same domain and range.
Deep property copy	Recursively apply shallow copy to all subproperties of a property.
Move Instance	Moves an instance from one concept to another.

Evo-
lutio
n

Representation → Semantics of change → Implementation → Propagation

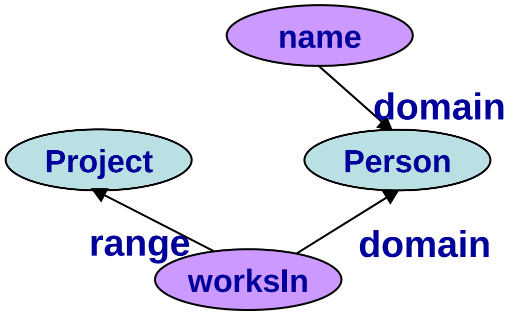
Ontology Evolution – Change representation

Composite change	Description
Move concept	Move a concept from one parent to another
Merge concepts	Replace several concepts with one and aggregate all instances.
	properties among them.
Extract superconcept	Create a common superconcept for a set of unrelated concepts and transfer common properties to it
Extract related concept	Extract related information into a new concept and relate it to the original concept.
Deep concept copy	Recursively apply shallow copy to all subconcepts of a concept.
Pull up properties	Move properties from a subconcept to a superconcept.
Pull down properties	Move properties from a superconcept to a subconcept.
Move properties	Move properties from one concept to another concept.
Shallow property copy	Duplicate a property with same domain and range.
Deep property copy	Recursively apply shallow copy to all subproperties of a property.
Move Instance	Moves an instance from one concept to another.

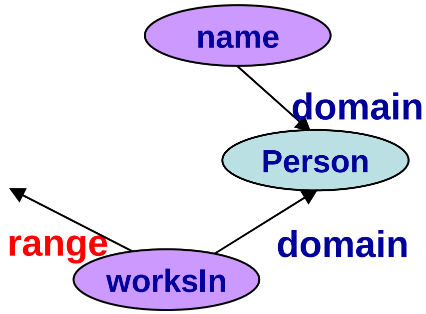


Ontology Evolution – Semantics of change

- Enables resolution of changes in a systematic manner, ensuring consistency of the whole ontology



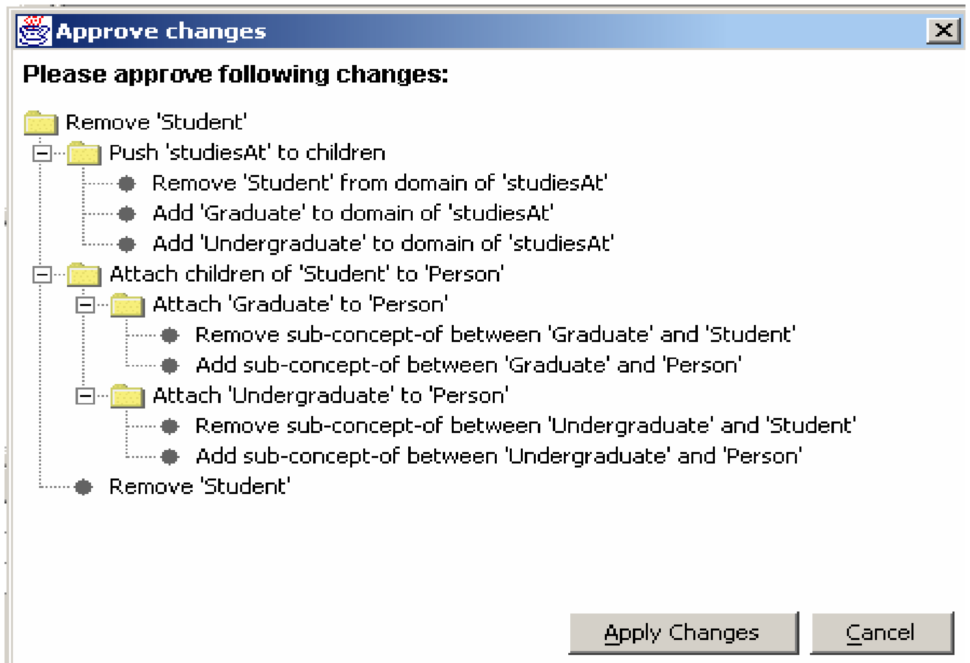
Tim:Person.
Tim[ID->711].



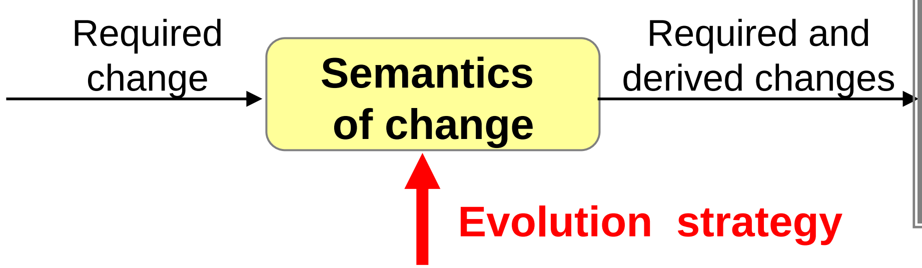
Ontology Evolution – Change implementation

- After user's approval all changes are applied to the ontology

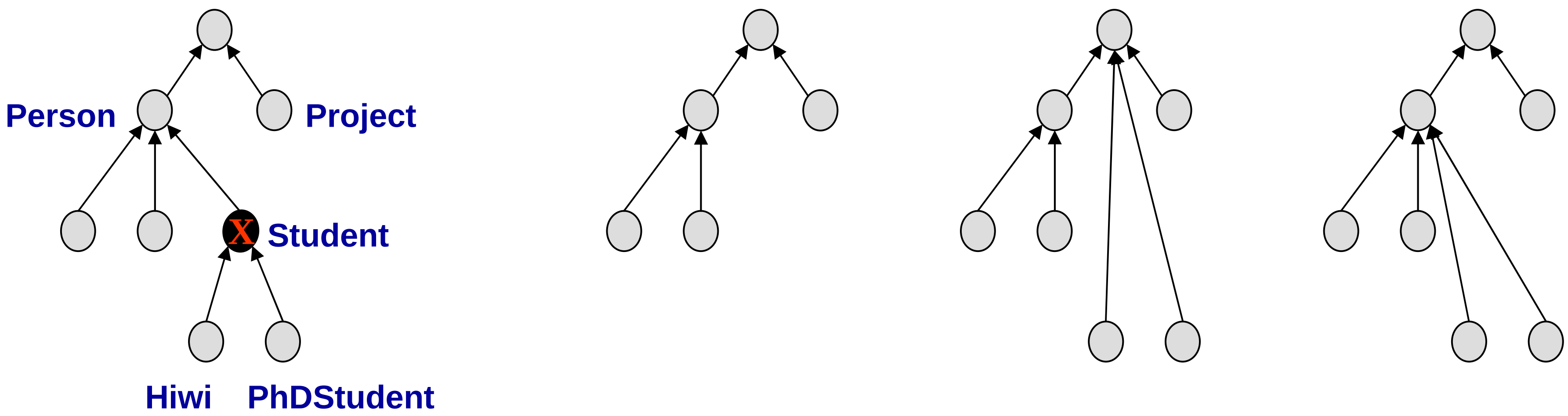
- Since it is necessary to perform several changes together, the transaction server is needed.



Evolution Strategies



An evolution strategy unambiguously defines the way how changes will be resolved



Evo-
lution

Evolution Strategies

Elementary evolution strategies

Resolution points:

- how to handle orphaned concepts;
- how to handle orphaned properties;

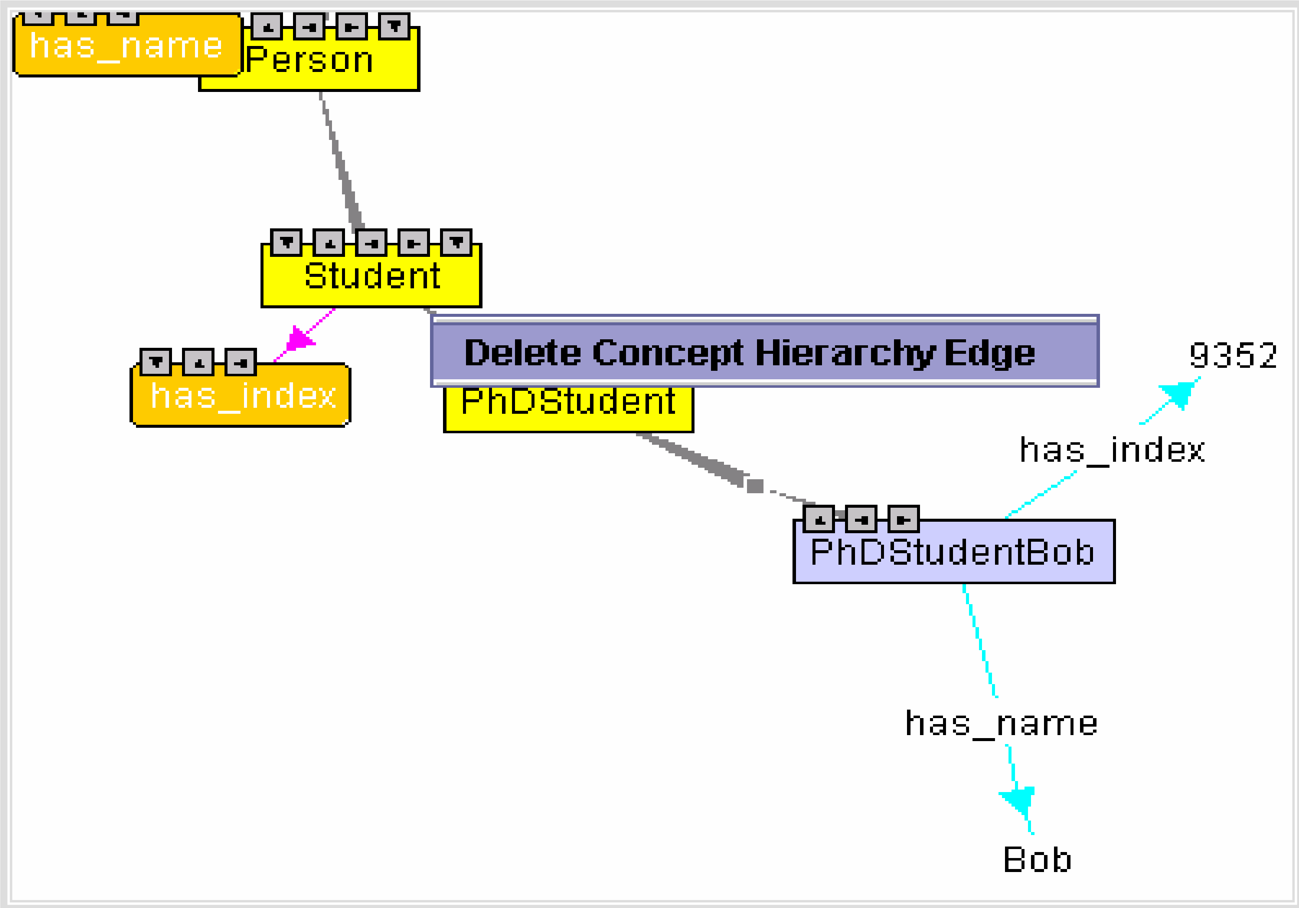
- delete
- reconnect to the root
- reconnect to the superconcepts

- how to propagate properties to the concept whose parent changes;
- what constitutes a valid domain of a property;
- what constitutes a valid range of a property;
- whether a domain (range) of a property can contain a concept that is at the same time a subconcept of some other domain (range) concept;
- the allowed shape of the concept hierarchy;
- the allowed shape of the property hierarchy;
- …

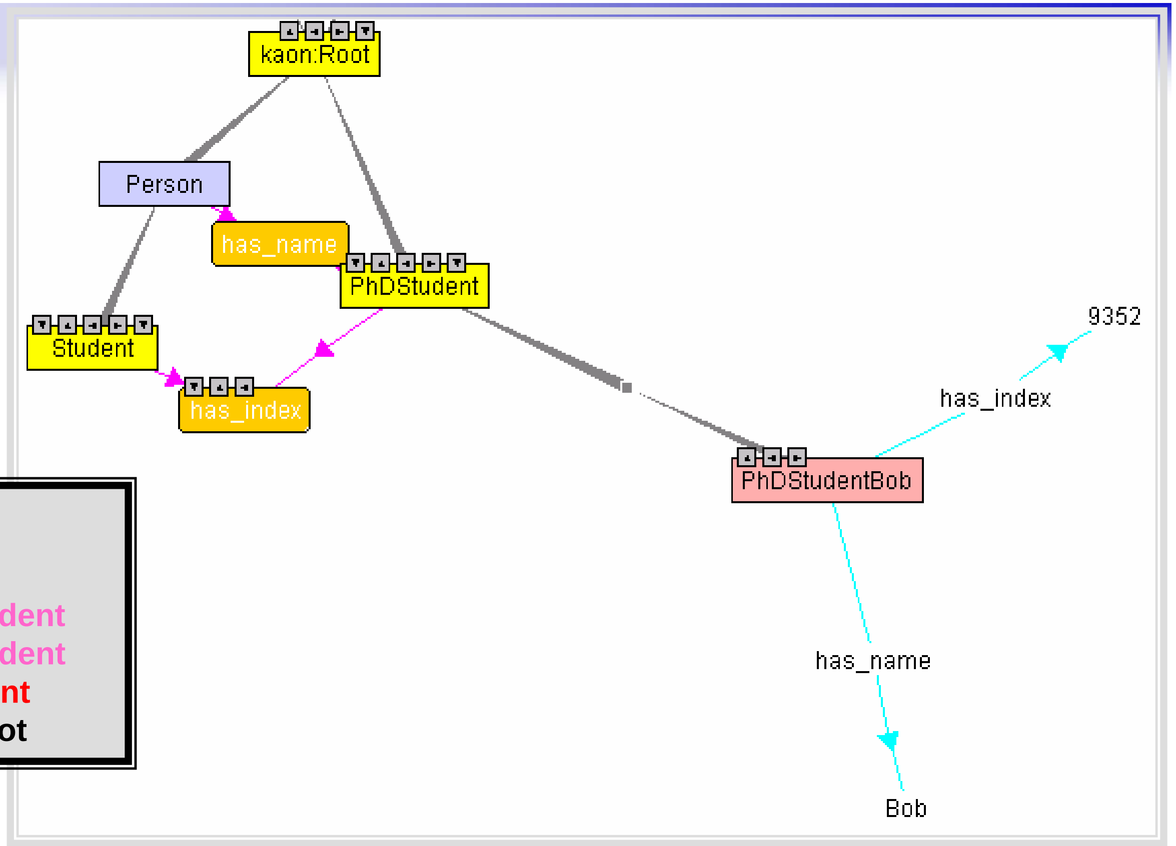
Common policy consisting of a set of elementary evolution strategies, each giving an answer for one resolution point, is an *evolution strategy*

Evo-
lution

Example

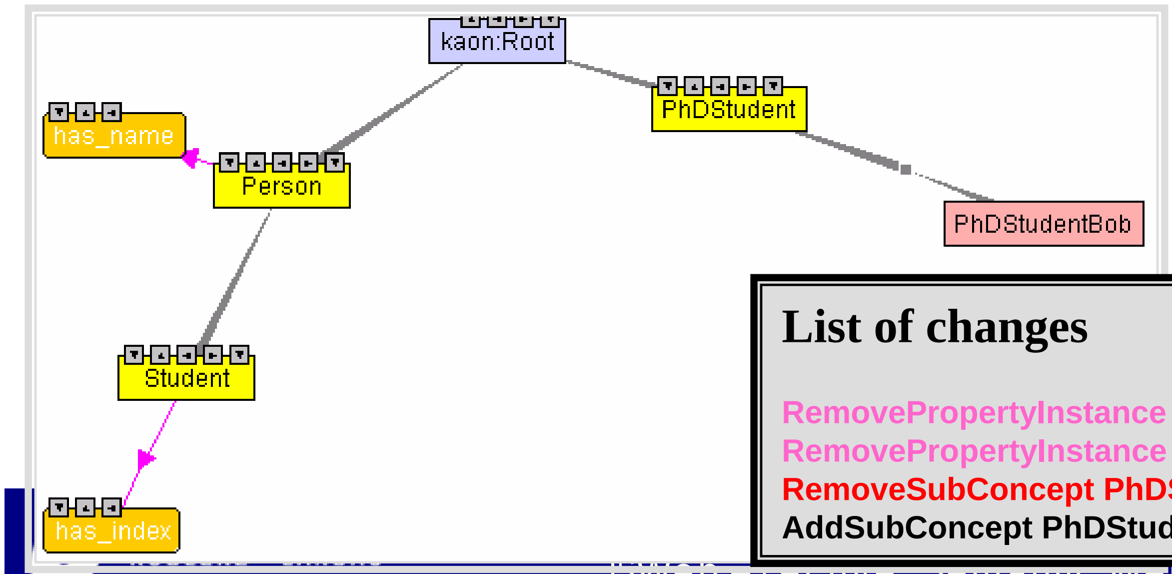


Evo-
lution



List of changes

AddPropertyDomain has_name, PhDStudent
AddPropertyDomain has_index, PhDStudent
RemoveSubConcept PhDStudent, Student
AddSubConcept PhDStudent, KAON:Root



List of changes

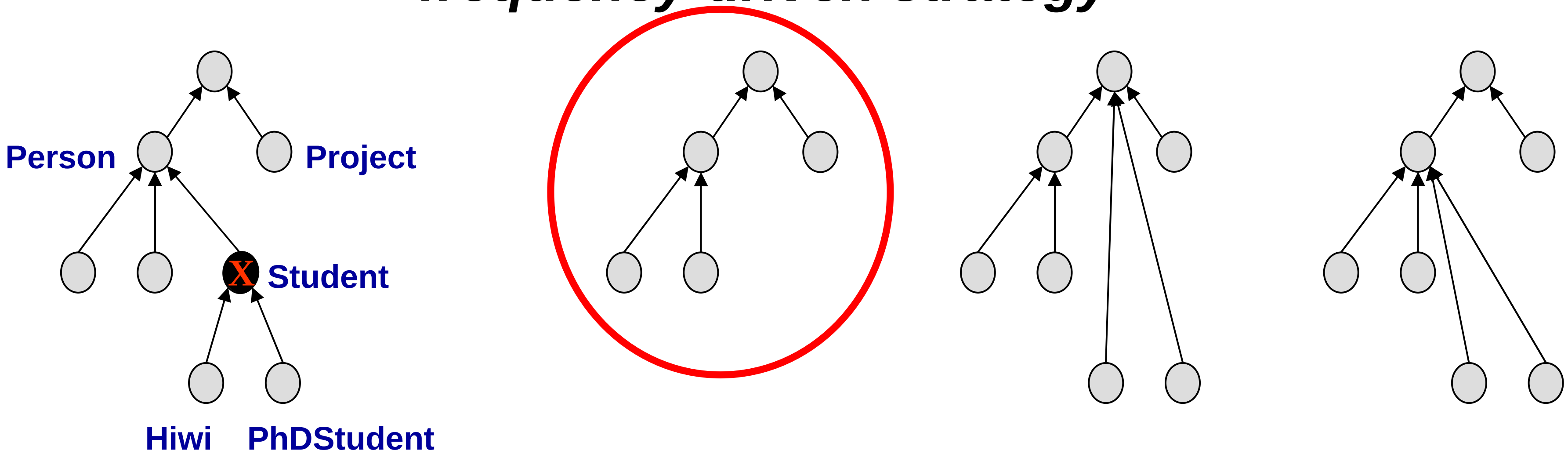
RemovePropertyInstance has_name, PhDStudentBob, Bob
RemovePropertyInstance has_index, PhDStudentBob, 9352
RemoveSubConcept PhDStudent, Student
AddSubConcept PhDStudent, KAON:Root

Evo-
lution

Advanced evolution strategies

Mechanism to prioritize and arbitrate among different evolution strategies, relieving the user of choosing them individually:

- ***structure-driven strategy***
- ***process-driven strategy***
- ***instance-driven strategy***
- ***frequency-driven strategy***



Evo-
lution

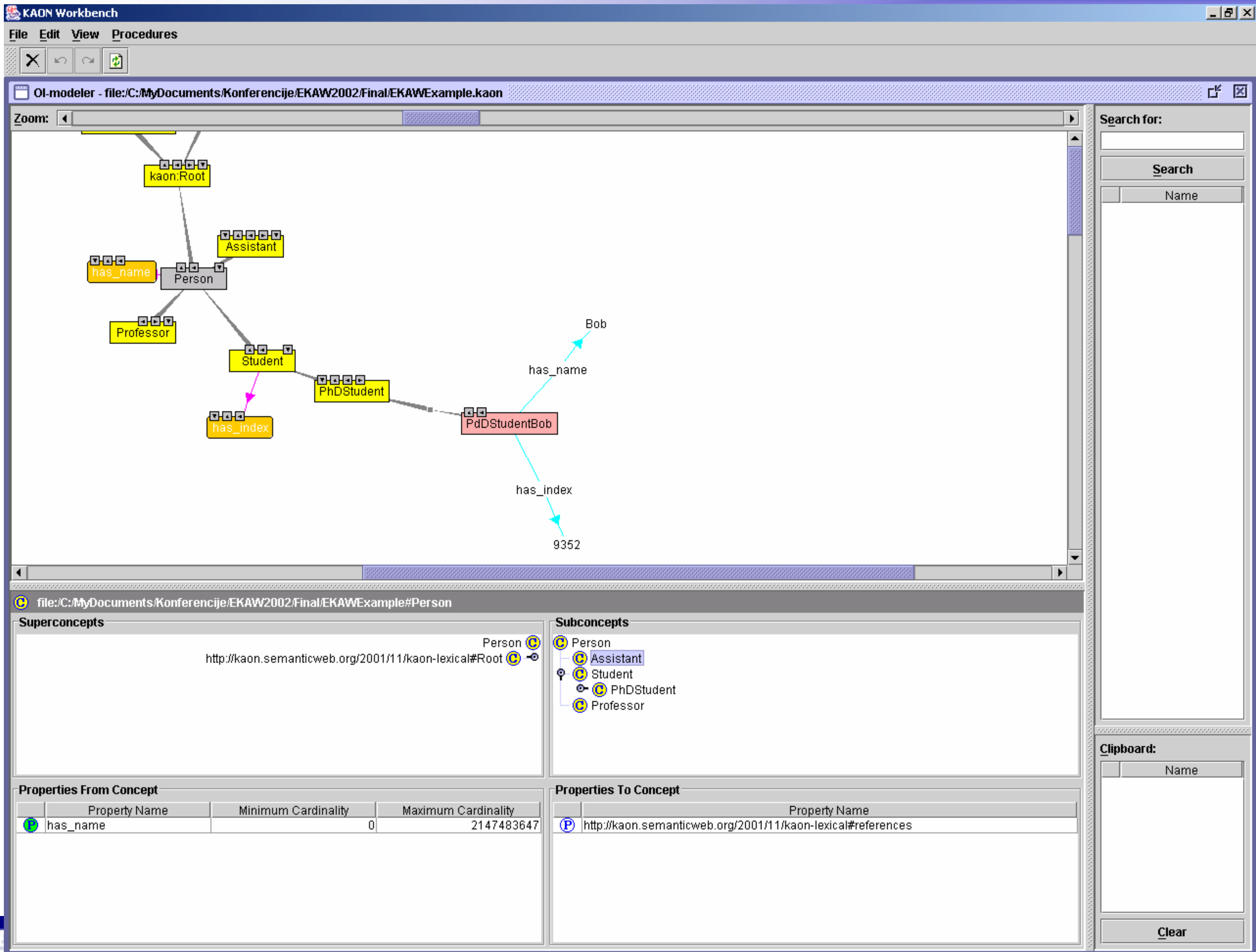
Implementation

http://kaon.semanticweb.org

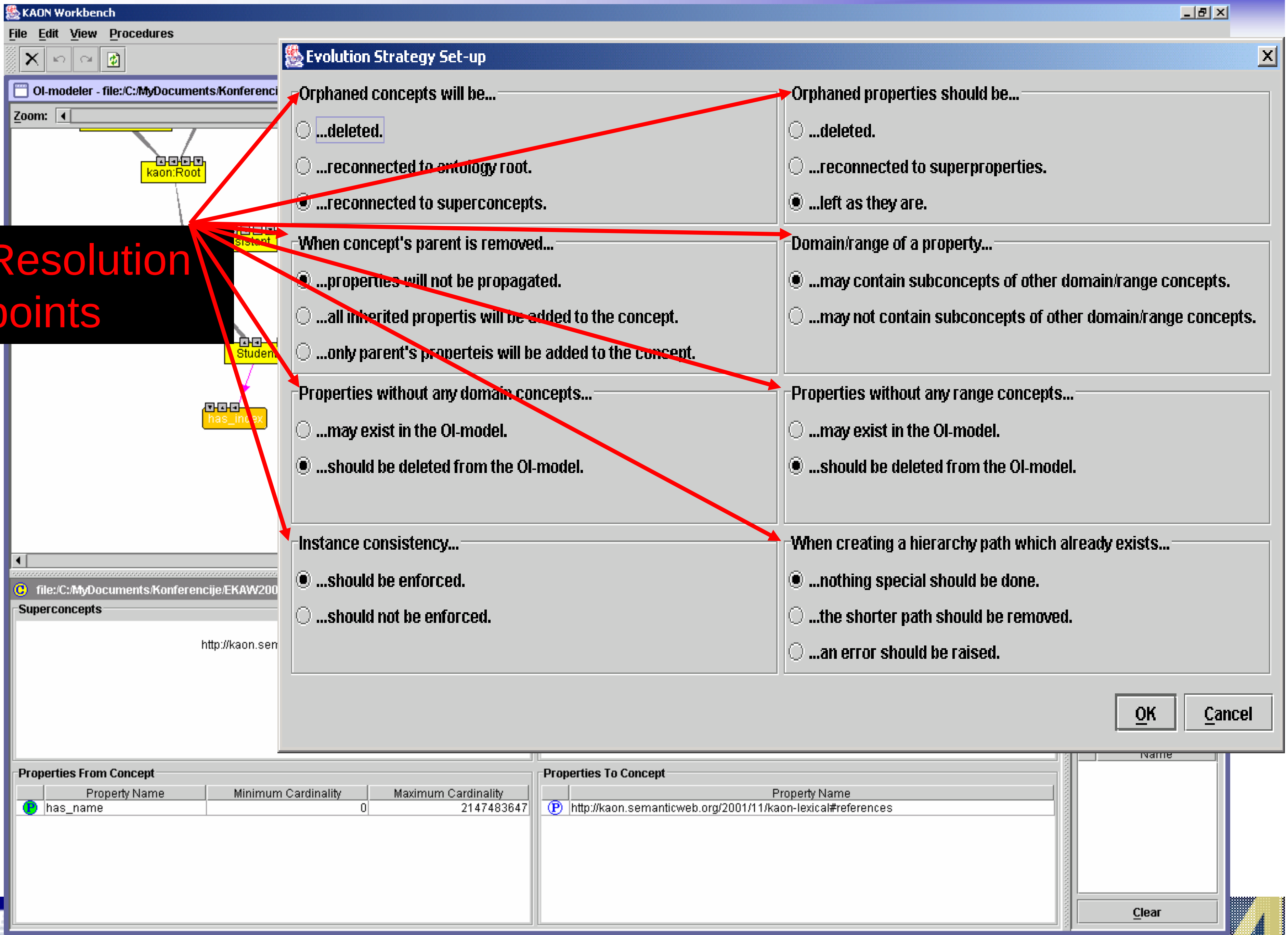
Applications & Services
- OIModeler - Ontology and Metadata Engineering Tool
- KAON Portal and other User Interface Applications and Services

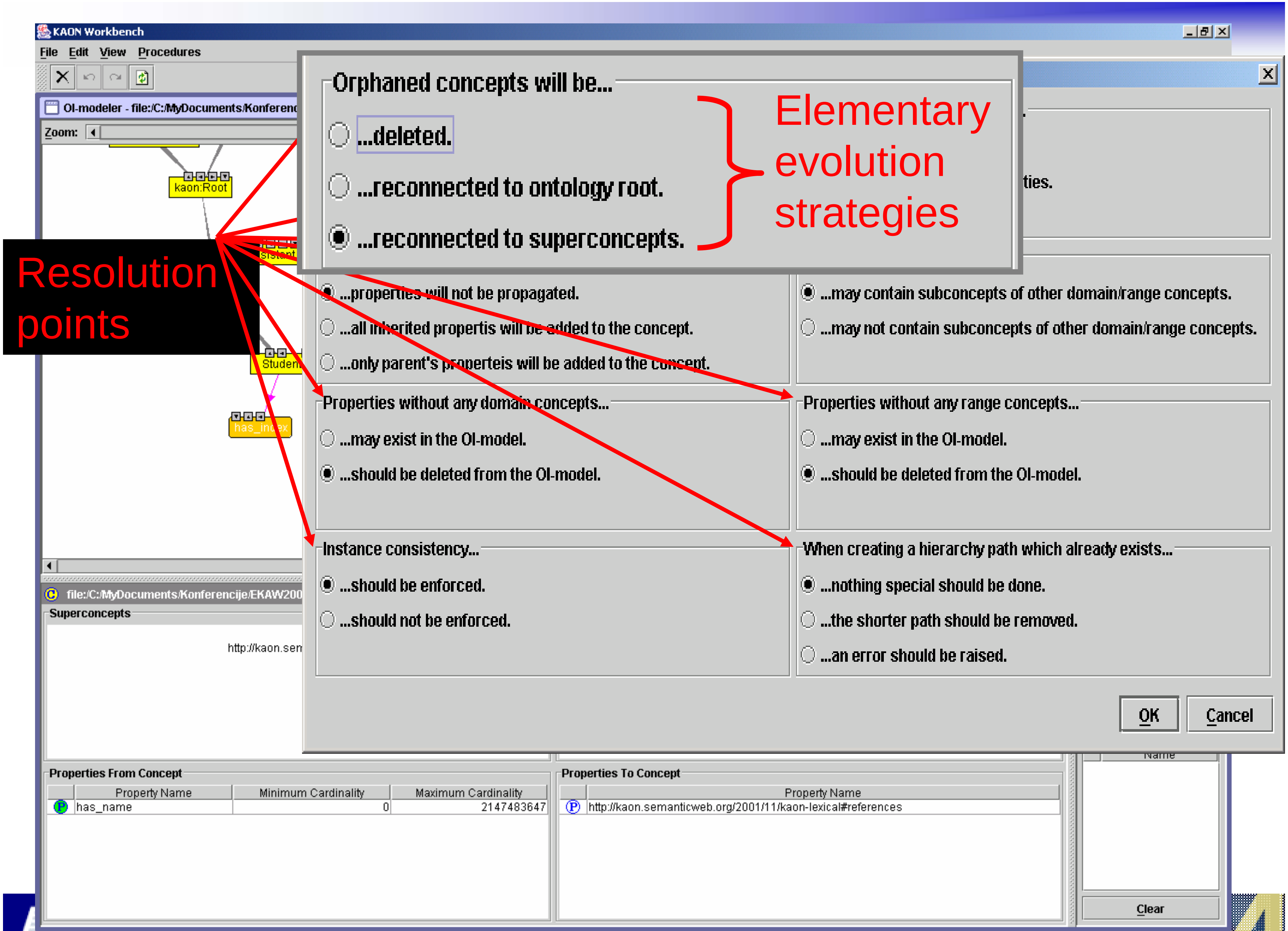
Middleware
- KAON Access Interface
- Change Discovery
- Interaction Logging
- Evolution Strategy
- Reversibility Services
- Evolution Logging
- KAON API
- RDF API
- KAON RDF Server

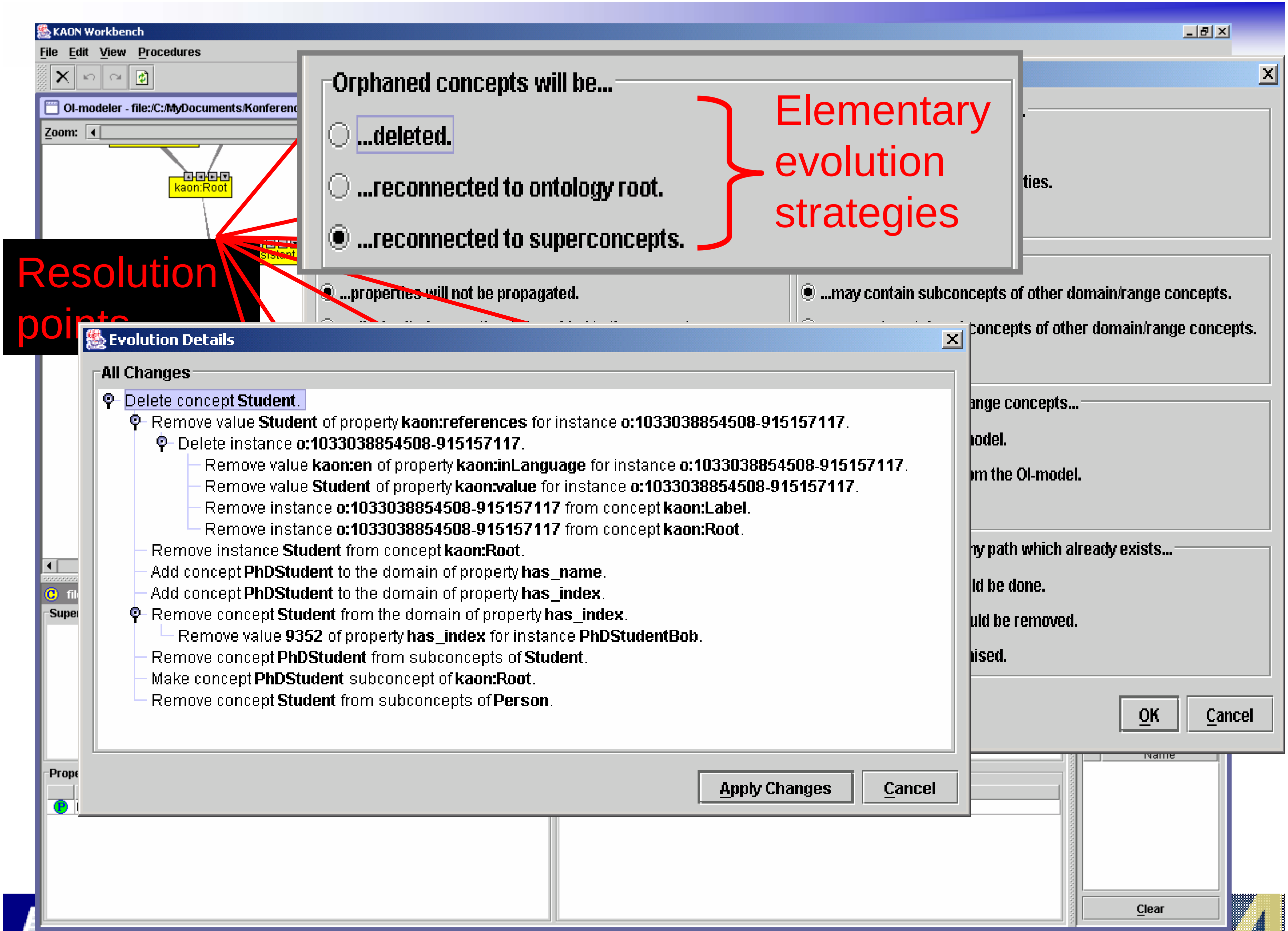
Data and Remote Services
- Persistence, Transactions, Security

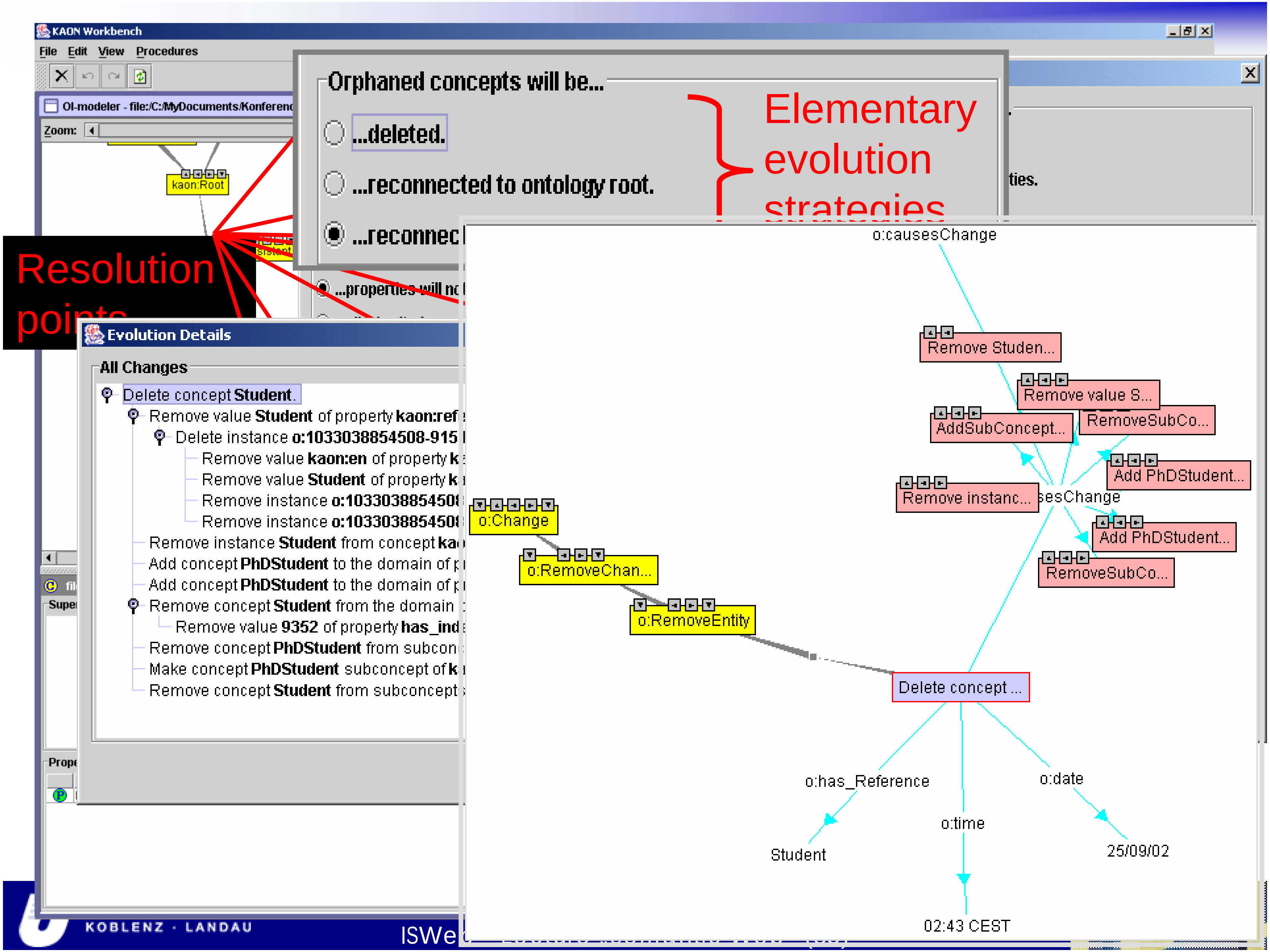

KAON Workbench
File Edit View Procedures
OI-modeler - file:/C:/MyDocuments/Konferencije/EKAW2002/Final/EKAWExample.kaon
Zoom:
kaon:Root
Assistant
has_name
Person
Professor
Student
PhDStudent
has_index
PdDStudentBob
Bob
has_name
has_index
9352
Search for:
Search
Name
file:/C:/MyDocuments/Konferencije/EKAW2002/Final/EKAWExample#Person
Superconcepts
Person
http://kaon.semanticweb.org/2001/11/kaon-lexical#Root
Subconcepts
Person
Assistant
Student
PhDStudent
Professor
Clipboard:
Name
Properties From Concept
Property Name
Minimum Cardinality
Maximum Cardinality
has_name
0
2147483647
Properties To Concept
Property Name
http://kaon.semanticweb.org/2001/11/kaon-lexical#references
Clear

Resolution points




KAON Workbench
File Edit View Procedures
OI-modeler - file:/C:/MyDocuments/Konferenc
Zoom:
kaon:Root
Studen
has_index
file:/C:/MyDocuments/Konferencije/EKAW200
Superconcepts
http://kaon.sem
Resolution points
Orphaned concepts will be...
...deleted.
...reconnected to ontology root.
...reconnected to superconcepts.
Elementary evolution strategies
ties.
...properties will not be propagated.
...all inherited propertis will be added to the concept.
...only parent's properteis will be added to the concept.
...may contain subconcepts of other domain/range concepts.
...may not contain subconcepts of other domain/range concepts.
Properties without any domain concepts...
...may exist in the OI-model.
...should be deleted from the OI-model.
Properties without any range concepts...
...may exist in the OI-model.
...should be deleted from the OI-model.
Instance consistency...
...should be enforced.
...should not be enforced.
When creating a hierarchy path which already exists...
...nothing special should be done.
...the shorter path should be removed.
...an error should be raised.
OK
Cancel
Properties From Concept
Property Name
Minimum Cardinality
Maximum Cardinality
has_name
0
2147483647
Properties To Concept
Property Name
http://kaon.semanticweb.org/2001/11/kaon-lexical#references
Name
Clear


KAON Workbench
File Edit View Procedures
OI-modeler - file:/C:/MyDocuments/Konferenc
Zoom:
kaon:Root
Orphaned concepts will be...
...deleted.
...reconnected to ontology root.
...reconnected to superconcepts.
Elementary evolution strategies
Resolution points
...properties will not be propagated.
...may contain subconcepts of other domain/range concepts.
concepts of other domain/range concepts.
ties.
Evolution Details
All Changes
Delete concept Student.
Remove value Student of property kaon:references for instance o:1033038854508-915157117.
Delete instance o:1033038854508-915157117.
Remove value kaon:en of property kaon:inLanguage for instance o:1033038854508-915157117.
Remove value Student of property kaon:value for instance o:1033038854508-915157117.
Remove instance o:1033038854508-915157117 from concept kaon:Label.
Remove instance o:1033038854508-915157117 from concept kaon:Root.
Remove instance Student from concept kaon:Root.
Add concept PhDStudent to the domain of property has_name.
Add concept PhDStudent to the domain of property has_index.
Remove concept Student from the domain of property has_index.
Remove value 9352 of property has_index for instance PhDStudentBob.
Remove concept PhDStudent from subconcepts of Student.
Make concept PhDStudent subconcept of kaon:Root.
Remove concept Student from subconcepts of Person.
Apply Changes
Cancel
OK
Cancel
Clear


KAON Workbench
File Edit View Procedures
OI-modeler - file:/C:/MyDocuments/Konferenc
Zoom:
kaon:Root
Orphaned concepts will be...
...deleted.
...reconnected to ontology root.
...reconnec
...properties will no
Elementary evolution strategies
Resolution points
Evolution Details
All Changes
Delete concept Student.
Remove value Student of property kaon:ref
Delete instance o:1033038854508-915
Remove value kaon:en of property k
Remove value Student of property k
Remove instance o:103303885450
Remove instance o:103303885450
Remove instance Student from concept ka
Add concept PhDStudent to the domain of p
Add concept PhDStudent to the domain of p
Remove concept Student from the domain
Remove value 9352 of property has_ind
Remove concept PhDStudent from subcon
Make concept PhDStudent subconcept of k
Remove concept Student from subconcept
o:causesChange
Remove Studen...
Remove value S...
AddSubConcept...
RemoveSubCo...
Add PhDStudent...
Remove instanc...
sesChange
Add PhDStudent...
RemoveSubCo...
o:Change
o:RemoveChan...
o:RemoveEntity
Delete concept ...
o:has_Reference
o:time
o:date
Student
25/09/02
02:43 CEST
KOBLENZ · LANDAU
ISWe

Evo-
lution

Evolution wrap-up

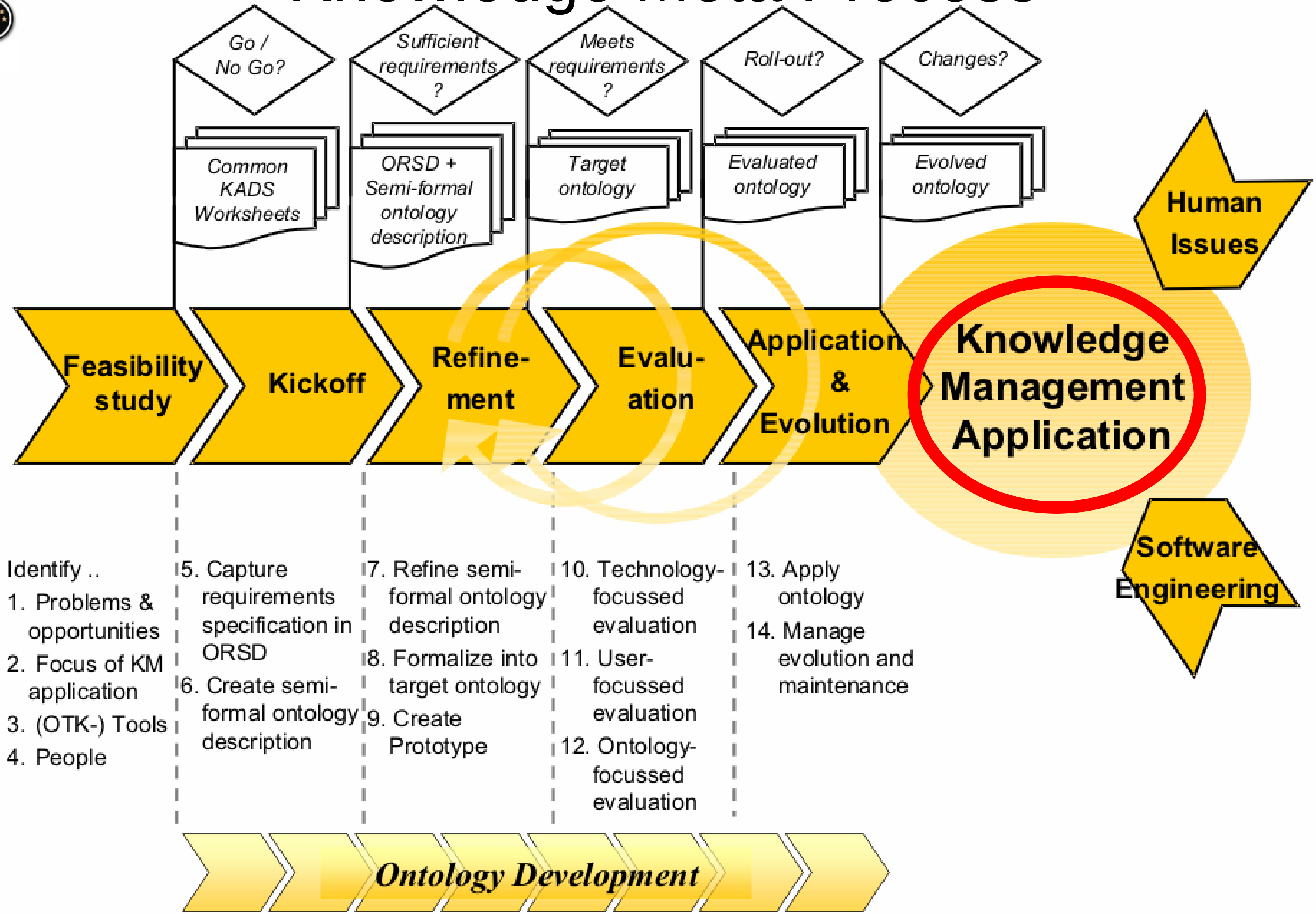
OntoLogging:

- process-based approach for ontology evolution
- Evolution strategies that enable the customisation of the ontology evolution process
- Implementation in KAON framework

Ongoing work:

- Evolution between distributed ontologies
- Change discovery

OTK Methodology:
Knowledge Meta Process

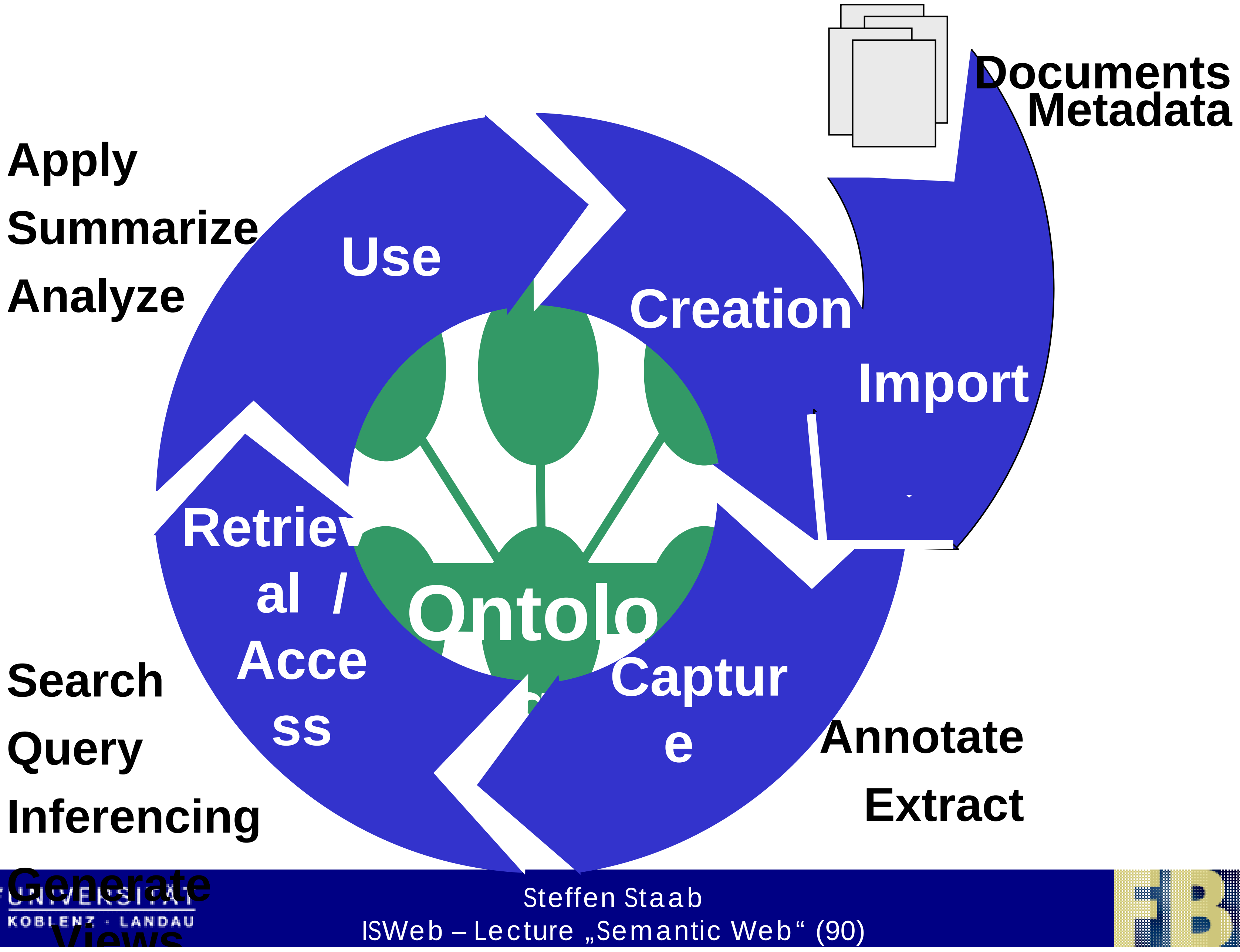


Conclusions on Knowledge Meta Process

Experiences from OTK Case Studies

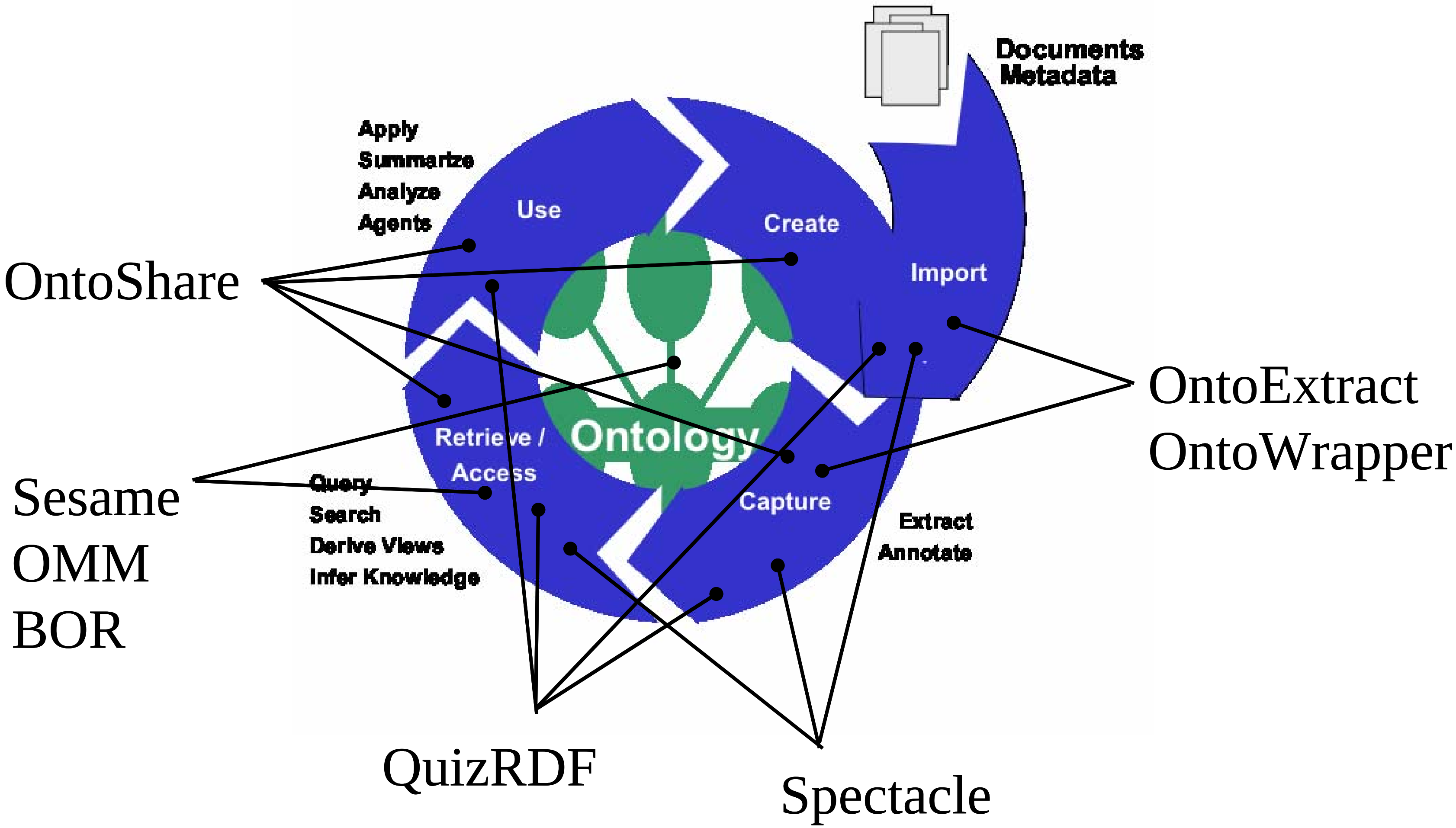
- **Guidelines** for domain experts from industry have to be pragmatic
  1. Train the user about ontologies
  2. Show the concrete advantage of the KMS
  3. Model precisely – but allow for imprecise views (most users cannot distinguish classes vs instances or isa vs partOf)
- **Plan for Maintenance**
- **Avoid/Reduce chicken-and-egg problem**
  1. Plan für content that makes KMS interesting
  2. Show quick win
- **Collaborative ontology engineering** requires sophisticated tool support *and* physical presence
- **Brainstorming** is a valuable add-on during the early stages of ontology engineering

Knowledge Process

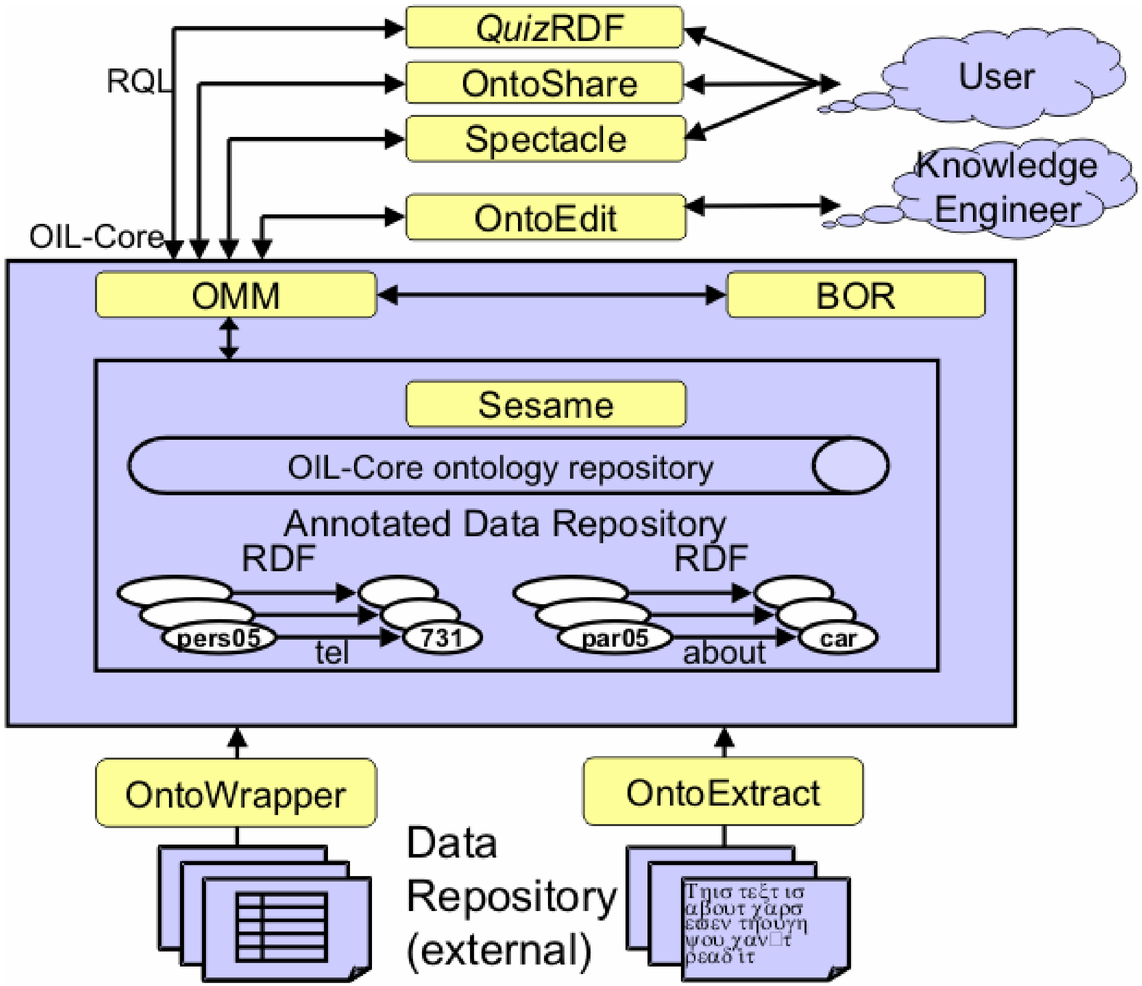


OTK Case Study @ BT

Users Portal



OTK Architecture



OTK Architecture