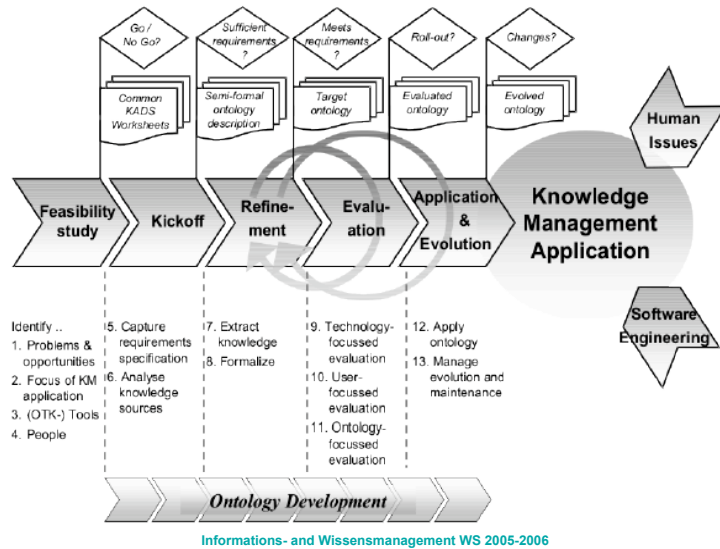


2.3.5 Methodology for Ontology Development



2.3.5.1 Kick-off Phase

- requirements are specified in the form of “**Ontology Requirements Specification Document**” (ORSD)
 - characterises the planned area of the ontology application
 - defines the scope of the ontology to be constructed
- knowledge sources are documents, databases, people, ... which can be consulted in order to make requirements specification
- Knowledge item analysis is important input (TM-2)
- result is **semiformal description of the ontology**

This phase can be treated as a process of making expert knowledge (more) explicit

Ontology Requirements Specification Document

ORSD is composed of different parts:

- **administrative information**
 - name
 - date
 - involved ontology engineer(s)
- **requirements specification**

ORSD - example

Ontology Requirements Specification Document	
Name:	skill-man-ontology
Date:	2001/03/22
Ontology Engineer:	T. Model
Domain and Goal:	
<ul style="list-style-type: none"> • The ontology is modeled for the domain <i>skill management</i> which is part of the <i>human resource development</i> • The ontology serves as a model for the knowledge distribution between HR department and project managers • Ontology serves as a base for semantic search for employee skills 	
Design Guidelines:	
The ontology contains lexical entries in the domain of human resource development. Skills in technologies and market domains of ABC have to be modeled in detail, the common business administration area has to be modeled on a more general level. Similar domain models contain about 400 – 600 lexical entries. Axioms are not planned yet.	
Supported Applications:	
Intranet based Skill Management System at company ABC	
Knowledge Sources:	
<ul style="list-style-type: none"> • HR department web pages • Handbook about employee development, internal document • Technology and product road map, internal document • Interviews with HR department and project managers 	
Users and Use cases:	
G. Peoplefind, Human Ressource Department; attached use case 1	
B. Boss, Project Manager attached use case 2	
Competency Questions:	
Attached CQ 1	

Requirements Specification (I)

■ Domain and Goal

- what is the objective of the planned KM application
- based on task analysis (TM-1)

■ Design Guidelines

- description of domain in use
- estimation of size of ontology
- exploit knowledge item worksheet (TM-2)

■ Supported Applications

- brief characteristics of planned application
- specification of system environment

Requirements Specification (II)

Knowledge Sources

■ types of knowledge sources may be very different

- domain experts
- (reusable) ontologies
- documents / systems
 - dictionaries
 - thesauri
 - product descriptions
 - organisational charts
 - employee role descriptions
 - ...

Requirements Specification (III)

■ Usage Scenarios (Users and Use cases)

- describe users/user groups
- identify stakeholders
- describe usage scenarios
 - how do they want to use the system?
 - what kind of support do they expect ?
 - use e.g. UML use-case diagrams

■ Competency Questions

- define collection of queries that should be supported by the system
- analyze queries to find relevant lexical entries (concepts and relations)
- explore scenarios
- collect competency questionnaire

Competency Questionnaire

Competency Questionnaire No. 1			
Name:		skill-man-ontology	
Date:		2001/03/22	
Ontology Engineer:		T. Model	Domain Expert: X. Pert
No.	Competency Question	Lexical Entries	Type
Q1	Which of our consultants has experience with JAVA programming language?	consultant	concept
		consultant <i>is a</i> employee	<i>isA</i> relation
		JAVA	concept
		programming language	concept
		JAVA <i>is a</i> programming language	<i>isA</i> relation
		programming language <i>is a</i> skill	<i>isA</i> relation
Q2	What is the salary of a senior programmer?	employee <i>has</i> experience with skill	relation
		salary	concept
Q3			

Initial lexicon - example

Competency Questionnaire No. 1			
Name:		skill-man-ontology	
Date:		2001/03/22	
Ontology Engineer: T. Model Domain Expert: X.Pert			
No.	Competency Question	Lexical Entries	Type
Q1	Which of our consultants has experience with JAVA programming language?	consultant	concept
		consultant is a employee	isA relation
		JAVA	concept
		programming language	concept
		JAVA is a programming language	isA relation
		programming language is a skill	isA relation
		employee has experience with skill	relation
Q2	What is the salary of a senior programmer?	salary	concept
Q3

lexical entries

potential concepts	potential relations
Consultant	HasExperienceWith
Employee	WorksIn
JAVA	Contains
Programming language	...
Experience	
Skill	
Programmer	
Project	
Customer	
Industry	
...	

Semiformal description of the ontology

- Develop initial model of the ontology
 - all necessary concepts, relations and axioms
- Semiformal description encompasses:
 - visual methods (e.g. UML)
 - textual descriptions

2.3.5.2 Refinement Phase

Construct a mature application-oriented ontology (formalise knowledge)

- Step 1: **Knowledge Extraction**
 - **top-down** (modeling concepts and relationships on a very generic level)
 - **bottom-up** (relevant concepts are extracted semi-automatically from available sources)
 - **middle-out** (identify the most important concepts which will then be used to obtain the remainder of the hierarchy by generalization and specialization)
- Step 2: **Formalise Ontology**
 - choose appropriate representation language,
 - e.g. Frame Logic, RDF Schema, OWL,...
 - decide which axioms are useful for the application

2.3.5.3 Evaluation Phase

- make a judgement of: (Gomez-Perez, 1996)
 - the ontologies,
 - their associated software environment and
 - documentation with respect to a frame of references
- feeds back to refinement phase
- need for a systematic approach for evaluation, which leads to the
 - consistent level of quality of the ontology and
 - acceptance by industry.

Evaluation Phase

Three types of the evaluation:

1. Technology-focused evaluation
 - properties of ontologies generated by development tools:
 - language conformity, consistency
 - technology properties:
 - interoperability, scalability, memory allocation, ...
2. User-focussed Evaluation
 - ontology requirements specification document
 - competency questions
 - usage patterns
3. Ontology-focussed Evaluation
 - formal evaluation methodologies for ontologies

2.3.5.4 Evolution Phase

Ontology evolution is the timely adaptation of an ontology as well as the consistent management/propagation of the changes to dependent artifacts

The variety of causes and consequences of the ontology changes makes ontology evolution a very complicated operation that should be realized as both technical and organizational **process**

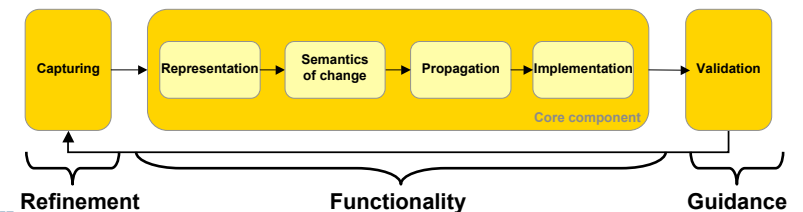
- What are the problems?
 - **Complexity** - ontology data models are rich
 - **Dependencies** - ontologies often reuse and extend other ontologies
 - **Physical distribution** - ontology development is a de-centralized and collaborative process
- ⇒ Needed are methods and tools for coping with changing dependent ontologies in a distributed environment

Evolution Phase

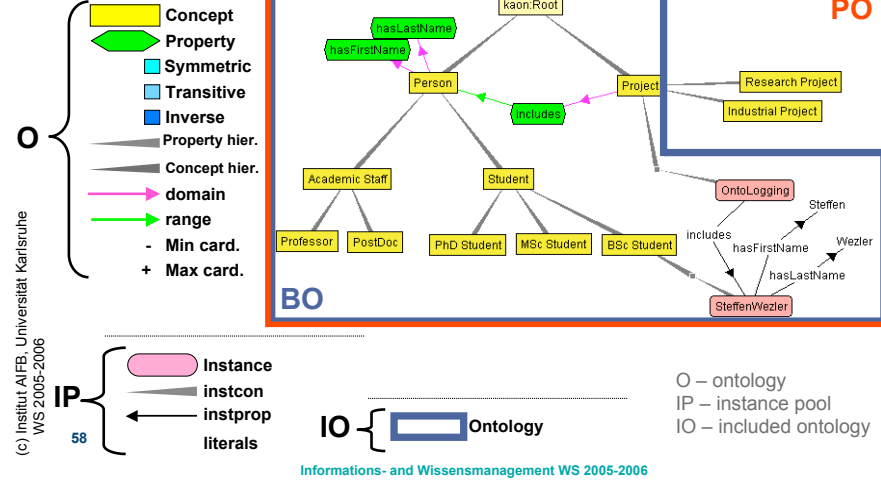
- running application has to adapt to changing environment
 - ontology has to adapt as well (**evolving ontology**)
- evolving aspects:
 - new lexical entries show up in application
 - extend ontology
 - lexical entries change meaning
 - change reference function
 - parts of ontology became obsolete (not needed anymore)
- set up clearly defined **organisational process** for updating the ontology
 - feeds back to refinement phase
 - update ontology by ontology engineer based on collection of proposed changes

Evolution Phase

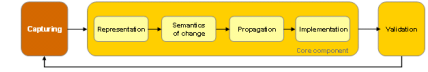
- Ontology evolution steps:
 - Change Capturing
 - Change Representation
 - Semantics of Change
 - Change Propagation
 - Change Implementation
 - Change Validation



An Ontology Example



1. Change Capturing



- Top-down changes are **explicit** changes
- Bottom-up changes are **implicit** changes
 - **Structure-driven** – exploits a set of heuristics to improve an ontology based on the analysis of its structure
 - **Data-driven** - detects the changes based on the analysis of the ontology instances
 - **Usage-driven** – takes into account the usage of the ontology

If all subconcepts have the same property, the property may be moved to the parent concept

1. Change Capturing



- Top-down changes are **explicit** changes
- Bottom-up changes are **implicit** changes
 - **Structure-driven** – exploits a set of heuristics to improve an ontology based on the analysis of its structure
 - **Data-driven** - detects the changes based on the analysis of the ontology instances
 - **Usage-driven** – takes into account the usage of the ontology

If no instance of a concept C uses any of the properties defined for C, but only properties inherited from the parent concept, we may assume that C is not necessary.

1. Change Capturing



- Top-down changes are **explicit** changes
- Bottom-up changes are **implicit** changes
 - **Structure-driven** – exploits a set of heuristics to improve an ontology based on the analysis of its structure
 - **Data-driven** - detects the changes based on the analysis of the ontology instances
 - **Usage-driven** – takes into account the usage of the ontology

By tracking when an entity has last been retrieved by a query, it may be possible to discover that some entities are out of date

Ontology Changes

Meta Entity \ Meta Changes	Add	Remove	
O	Concept	Add_Concept	Remove_Concept
	Property	Add_Property	Remove_Property
	Symmetric	Add_Symmetric	Remove_Symmetric
	Transitive	Add_Transitive	Remove_Transitive
	Inverse	Add_Inverse	Remove_Inverse
	Concept hierarchy	Add_SubConceptOf	Remove_SubConceptOf
	Property hierarchy	Add_SubPropertyOf	Remove_SubPropertyOf
	domain	Add_PropertyDomain	Remove_PropertyDomain
	range	Add_PropertyRange	Remove_PropertyRange
	Min cardinality	Add_MaxCard	Remove_MaxCard
Max cardinality	Add_MinCard	Remove_MinCard	
IP	Instance	Add_Instance	Remove_Instance
	instcon	Add_InstanceOf	Remove_InstanceOf
	instprop	Add_PropertyInstance	Remove_PropertyInstance
	literals	Add_AttributeInstance	Remove_AttributeInstance
	Ontology	Add_OIModel	Remove_OIModel

2. Change Representation

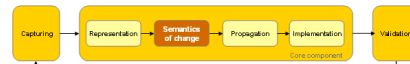


Three levels of changes:

- An **elementary change** is an ontology change that modifies *only one entity*
- A **composite change** is an ontology change that modifies *the neighbourhood* of an ontology entity
 - The neighbourhood of an entity consists of all entities that are directly linked to it
- A **complex change** is an ontology change that can be decomposed into *any combination* of at least two elementary and/or composite ontology changes



3. Semantics of Change

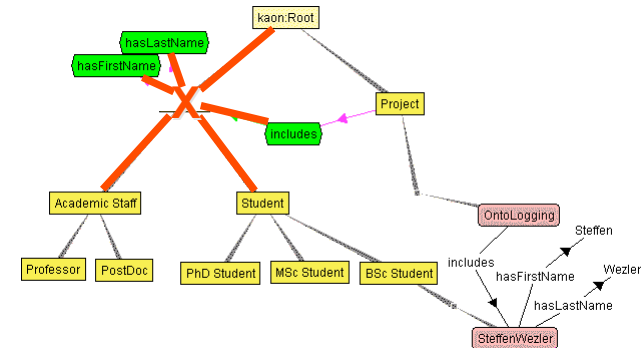


- The goal of this step is to resolve the required change in order to ensure **the ontology consistency**
- An ontology is **consistent** if it satisfies all the consistency constraints of the ontology model
 - **Invariants** are consistency rules that must hold for every ontology
 - **User-defined constraints** represent guidelines for building well-formed ontologies
 - **Soft-constraints** can be temporarily invalidated in order to make the ontology evolution easier

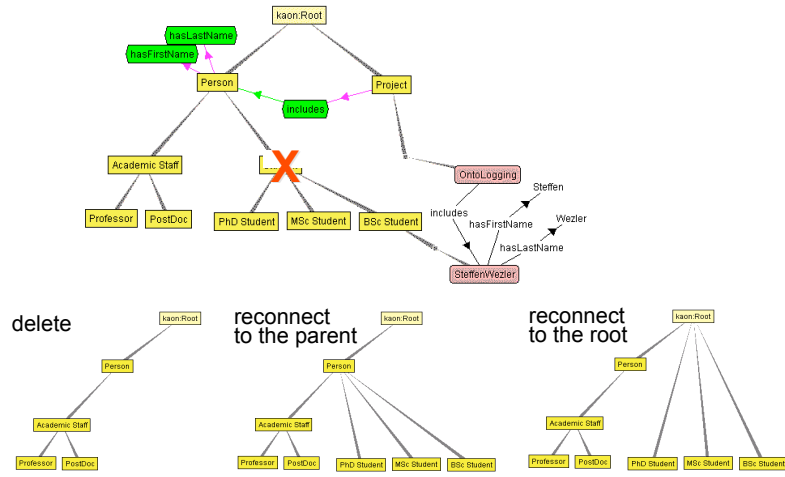
Semantics of change



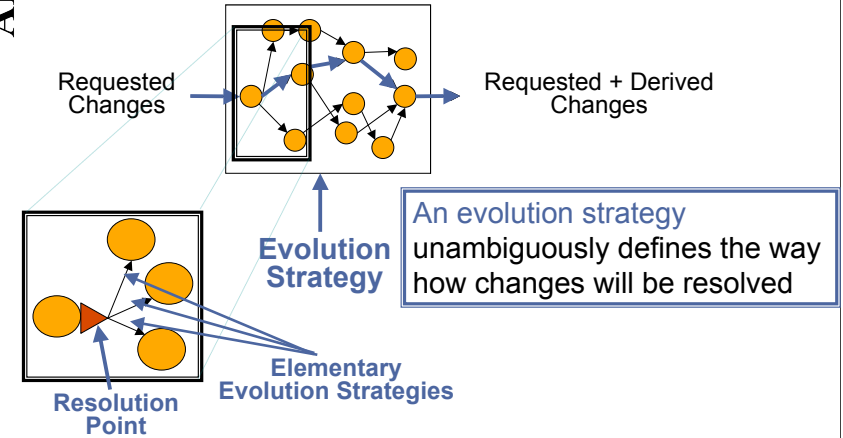
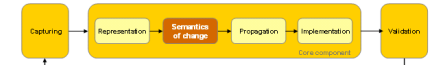
- Enables resolution of changes in a systematic manner by ensuring consistency of the whole ontology



Semantics of change



Evolution Strategies



Evolution Strategies



Resolution points:

- how to handle orphaned concepts
 - delete
- how to handle orphaned properties
 - reconnect to the superconcepts
 - reconnect to the root
- how to propagate properties to the concept whose parent has been changed
 - don't propagate any properties
 - propagate all properties
 - propagate only properties of the parent concept
- how to handle instances whose concept is deleted
- what constitutes a valid domain/range of a property
- ..

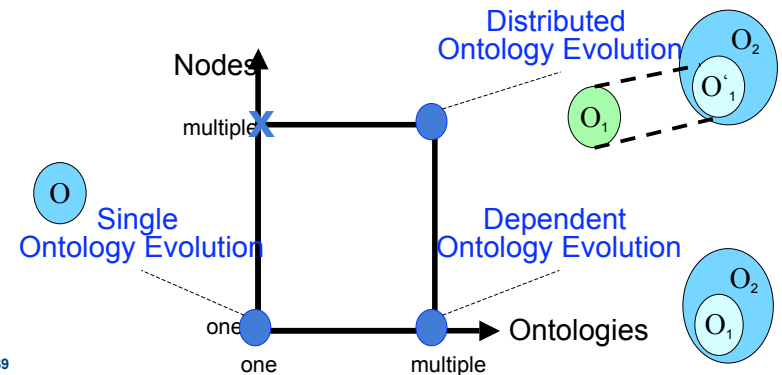
Advanced evolution strategies relieve an ontology engineer of choosing elementary evolution strategies individually

- structure-driven
- process-driven
- frequency-driven

4. Change Propagation

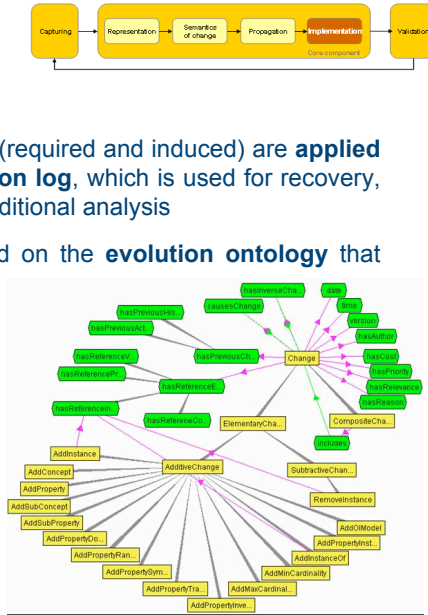


- The task of the change propagation phase is to bring **all dependent artefacts** automatically into a consistent state after an ontology update has been resolved



5. Change Implementation

- In this phase all changes (required and induced) are **applied** and tracked in an **evolution log**, which is used for recovery, change propagation or additional analysis
- An evolution log is based on the **evolution ontology** that enables
 - the representation
 - the analysis
 - the reasoning about
 - the realisation
 - the sharing
 - ...of ontological changes



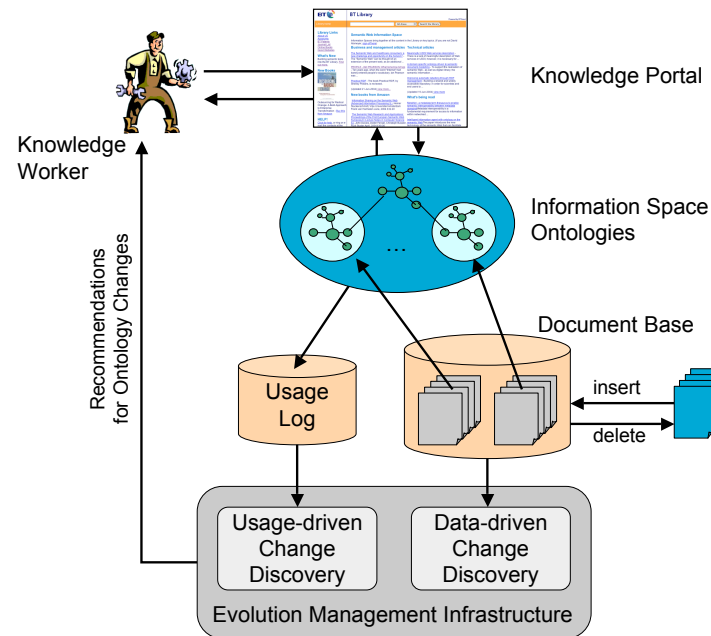
Informations- and Wissensmanagement WS 2005-2006

6. Change Validation

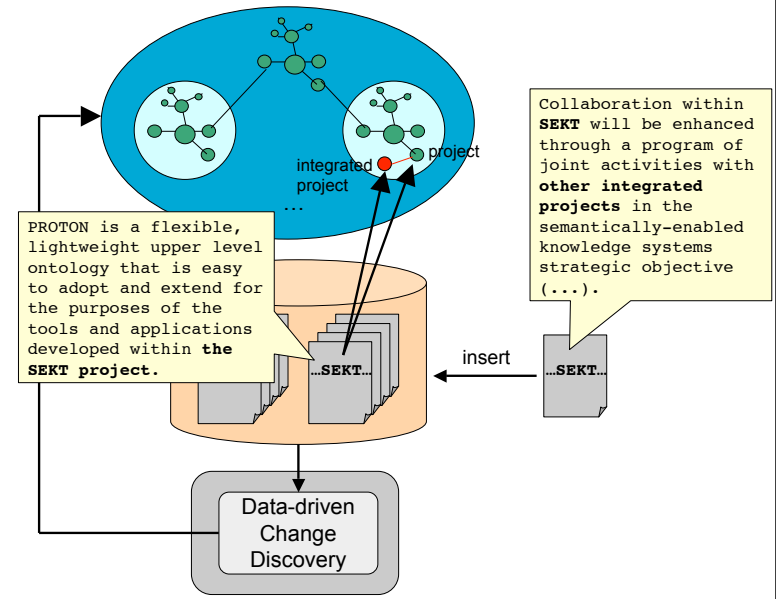
- The task of the change validation phase is to help ontology engineers find out whether they have built **the right ontology**
- It is done by:
 - presenting **information** needed to control evolution
 - when something goes wrong, explaining the situation in adequate detail and help the user resolve the problem
 - providing **reversibility** support
 - undoing all effects of some change



Informations- and Wissensmanagement WS 2005-2006



Informations- and Wissensmanagement WS 2005-2006

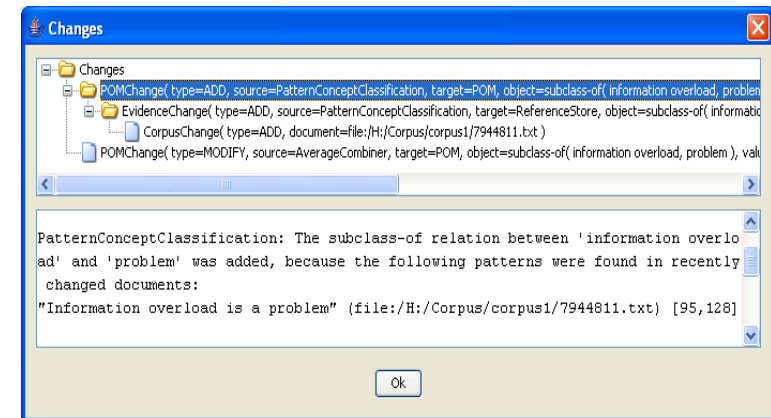


Informations- and Wissensmanagement WS 2005-2006

Change Discovery in Text2Onto

- Data-driven Change Discovery
 - Deduction of ontology changes from changes to the data
- Incremental Ontology Learning
 - Update **evidence** for ontology elements based on observed **corpus changes**
 - Generate suggestions (and **explanations**) for **ontology changes** based on new evidence
- Ontology Change Strategies
 - How are different types of **ontology elements** affected by particular changes to the corpus?

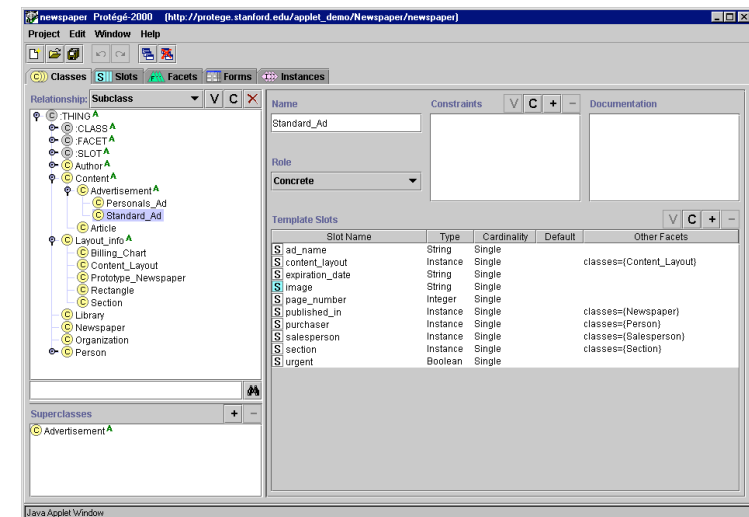
Data-driven Change Discovery



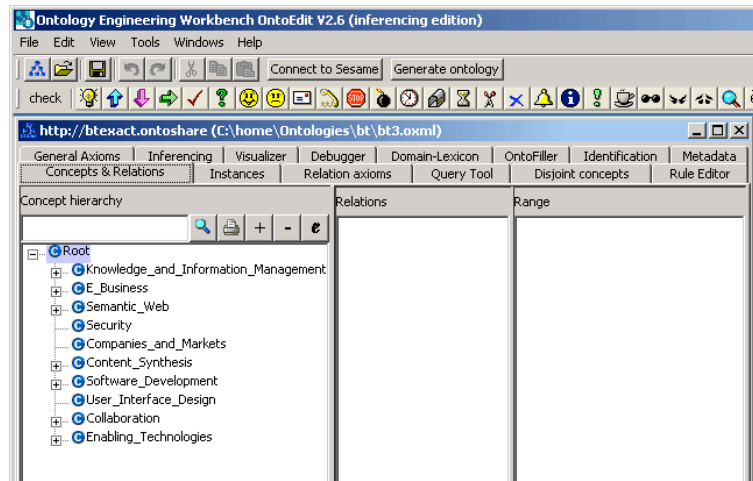
Software environment

- Ontology development has to be supported by suitable software environment
 - **graphical** interface
 - **abstract** representation language (as abstract as possible)
 - **distributed** development by **several** persons
 - aspects of Computer-Supported Co-operative Work
 - **export** in suitable representation languages
 - Frame Logic, RDF Schema, OWL
 - **import** of ontologies

Protege-2000 Ontologic-Editor



OntoEdit (Ontoprise)



2.4 Zusammenfassung

- Ontologien stellen semantische Informationen über
 - **Commonsense** Wissen (Zeit, Raum, ...) und über
 - **Anwendungsbereiche** bereit
- Ontologien geben Hinweis auf **relevante** Begriffe, Attribute, Beziehungen beim Aufbau domänenspezifischer Wissensbasen bzw. konzeptueller Schemabeschreibungen
 - Beziehungen führen zu **weiteren** relevanten Begriffen
 - Attribute innerhalb einer Begriffsdefinition kennzeichnen **zusätzliche relevante Merkmale** für domänenspezifische Begriffe
- Ontologien stellen Grundlage für semantisch-basierte Kommunikation bereit und sind deswegen auch relevant im Kontext von **WWW- bzw. Intranet-Suchmaschinen, E-Commerce, Web Applikationen und Wissensmanagement**
- Konstruktion einer Ontologie geschieht evtl. über Kombination einer Vielzahl anderer Ontologien
- Merging/Aligning von Ontologien ist in vielen Anwendungskontexten von Bedeutung

Prüfungsanmeldung ...bis 3. Feb. 2006!

Die Anmeldung erfolgt wie gewohnt durch **beides**:

1. Anmeldung am Rechner gegenüber von Raum 221 am AIFB
2. Einwurf des vollständig ausgefüllten Zulassungsscheins vom Prüfungsamt in den Briefkasten vor Raum 228 am AIFB

Bei Problemen: Michael Decker (Raum 228, Tel.: 608-4061)

Prüfungstermin: 22. Februar 2006

Weitere Infos zu den Klausuren gibt es unter
<http://www.aifb.uni-karlsruhe.de/StudiumUndPruefung/>