Janus:
Automatic Ontology Builder From XSD Files

Ivan BEDINI — Orange Labs

Benjamin NGUYEN, Georges GARDARIN — University of Versailles

UNIVERSITE DE VERSAILLES
SAINT-QUENTIN-EN-YVELINES

WWW 2008

orange

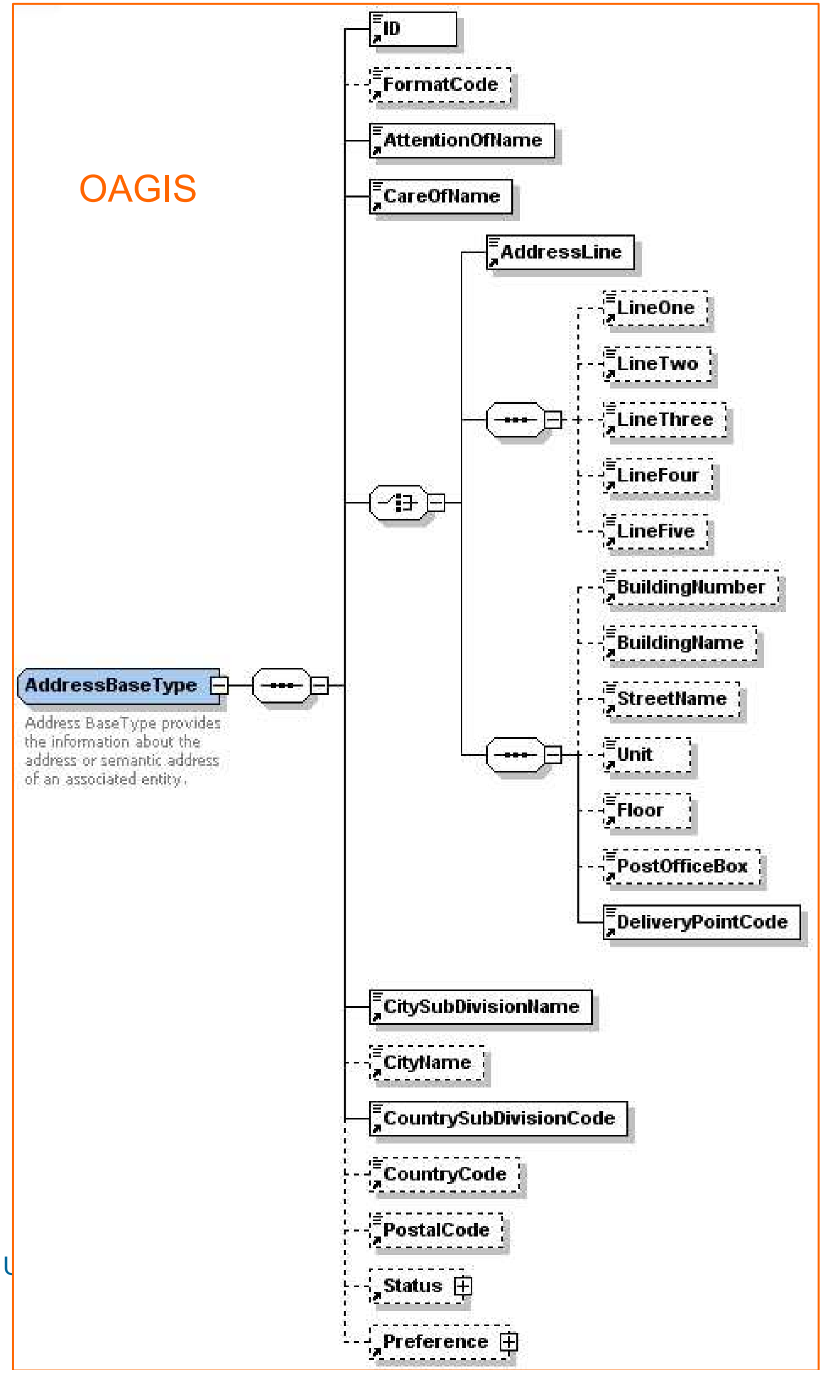
content

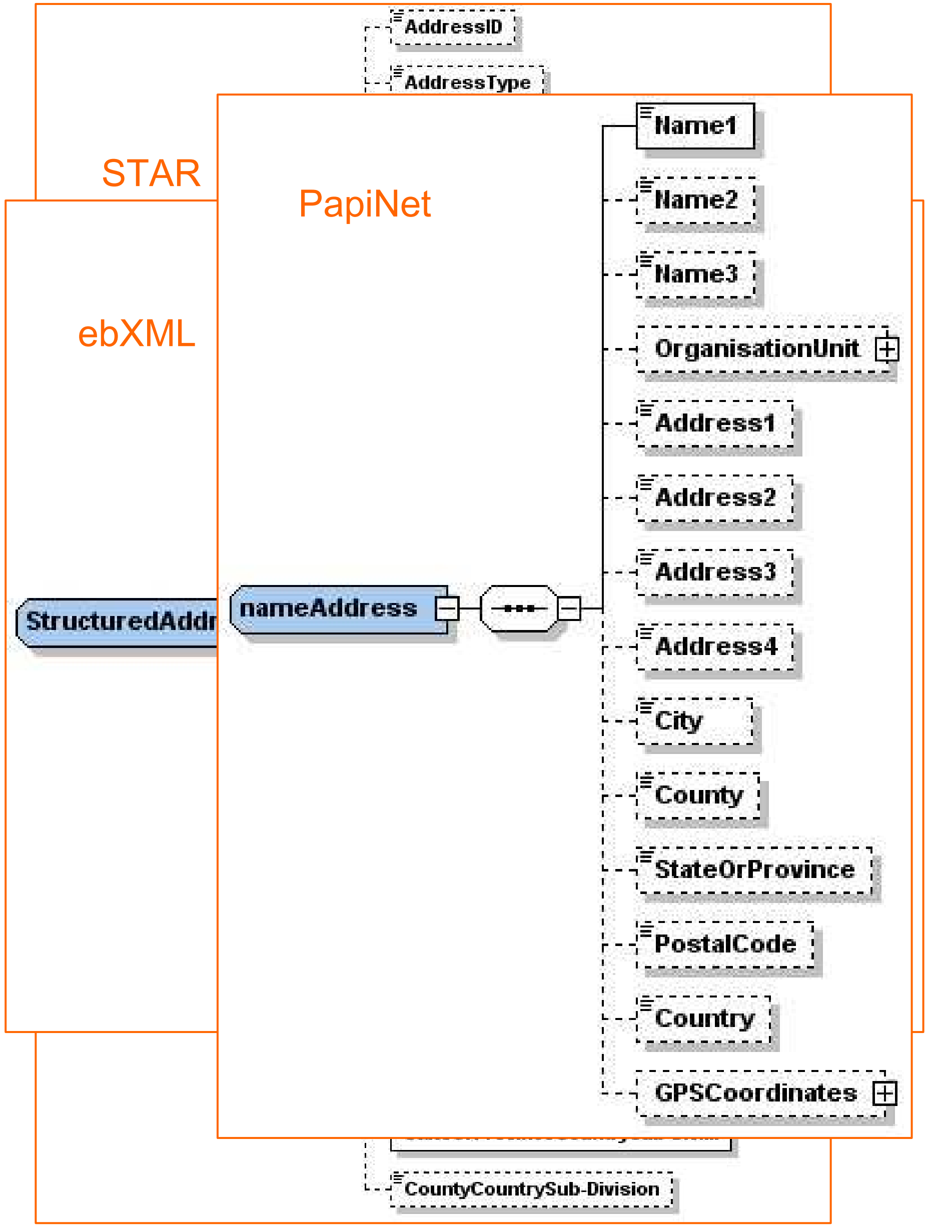
- B2B Use Case Challenge and Motivations
- Ontology Building Tools: Automation Approaches
- Ontology Building Methodology
- Janus: Automatic Ontology Builder Tool

B2B Use Case Challenge

- B2B Use Case
  - 75% of business exchanges declare implementing applications based on B2B standards (E-Business W@tch, 2007)
  - B2B bodies produce messages data definition by business area (Tourist, Retail, Insurance, Financial, Chemical, …), thus often we have different designs and ways of structuring the same set of concepts
  - We have investigated more than 30 B2B standards and
    - All of them provide XML based standards like XSD and DTD (we collected already ~3000 files)
    - Anyone officially provides ontology for business exchange data definition

- Challenge
  - XML documents provide likely annotated text with important information about objects and their structures
  - Schemas are built in a domain before ontologies and they are somehow related
  - More than one file to describe a domain and more domains to integrate on the fly and evolutive


OAGIS
ID
FormatCode
AttentionOfName
CareOfName
AddressLine
LineOne
LineTwo
LineThree
LineFour
LineFive
BuildingNumber
BuildingName
StreetName
Unit
Floor
PostOfficeBox
DeliveryPointCode
AddressBaseType
Address BaseType provides the information about the address or semantic address of an associated entity.
CitySubDivisionName
CityName
CountrySubDivisionCode
CountryCode
PostalCode
Status
Preference


STAR
AddressID
AddressType
ebXML
StructuredAddr
CountyCountrySub-Division
PapiNet
nameAddress
Name1
Name2
Name3
OrganisationUnit
Address1
Address2
Address3
Address4
City
County
StateOrProvince
PostalCode
Country
GPSCoordinates

Why Yet Another Tool?

- Manual generation of Ontologies is a strong task
  - How to manage "on the fly" integration?
  - How to manage evolution of concepts?
  - How to manage thousands of concepts?
  - Needs domain experts

- Automation is still limited
  - Alignment and merging of sources are complex and requires external knowledge not always available
  - Algorithms for concepts similarities discovery are computational time consuming
  - Multi-ontologies inputs are not treated. Existing tools mainly consider two ontologies at a time

- There are few tools for Ontology Learning from XML files

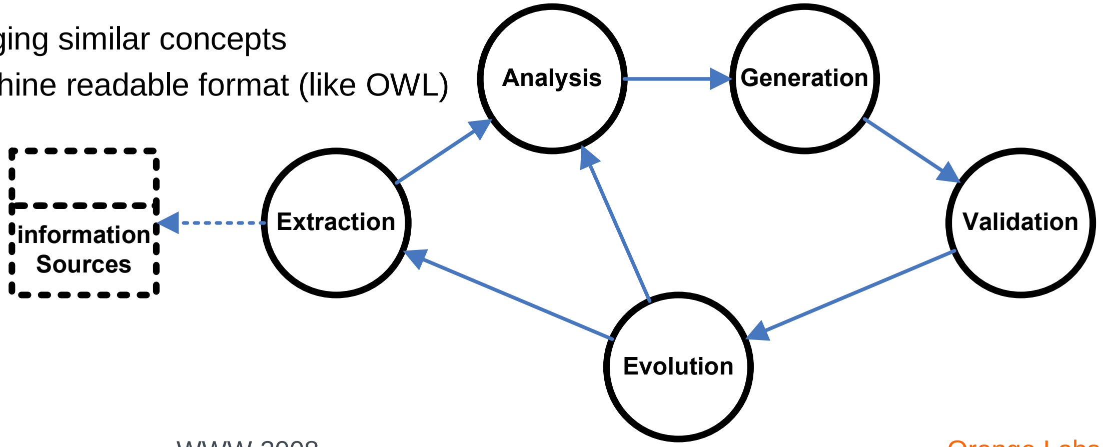
Automation of Ontology Building Approaches

- Conversion or translation from other formats (like ER Schemas, UML and XML Schemas)
  - Mainly XSL Transformations
  - Requires well defined and complete input source for the domain
  - High automation degree, but does not "elaborate" source information (e.g.: *WorkProgrConstrContract* becomes a concept of the ontology)
- Mining based
  - Mainly from free text input sources with NLP (Natural Language Process) techniques
  - Requires a lot of human assistance or of a reference ontology for the domain
- External knowledge based
  - Normally used to build or enrich a domain ontology
  - A set of words is provided as input and external resources like WordNet, the WWW or an existing reference ontology to get more information
  - The automation is good enough but requires a reference knowledge of the domain
- Frameworks
  - This modular approach to the generation provides better results then previous
  - Modules integration is often human
  - Input is often binary (e.g.: 2 XML files or 2 ontologies at a time)

Ontology Building Methodology

- Our methodology provides a general view of the automation aspect of the ontology generation. It does not target ontology engeeners.
- Given an input source the Ontology Learning and generation process is composed by the following steps:

1. Extraction
   - Knowledge retrieval and Normalization
2. Analysis
   - Define classes, properties and data-type
   - Build semantic networks of concepts (define similarities)
3. Generation
   - Produce a global view by merging similar concepts
   - Provide transformation to machine readable format (like OWL)
4. Validation
5. Evolution

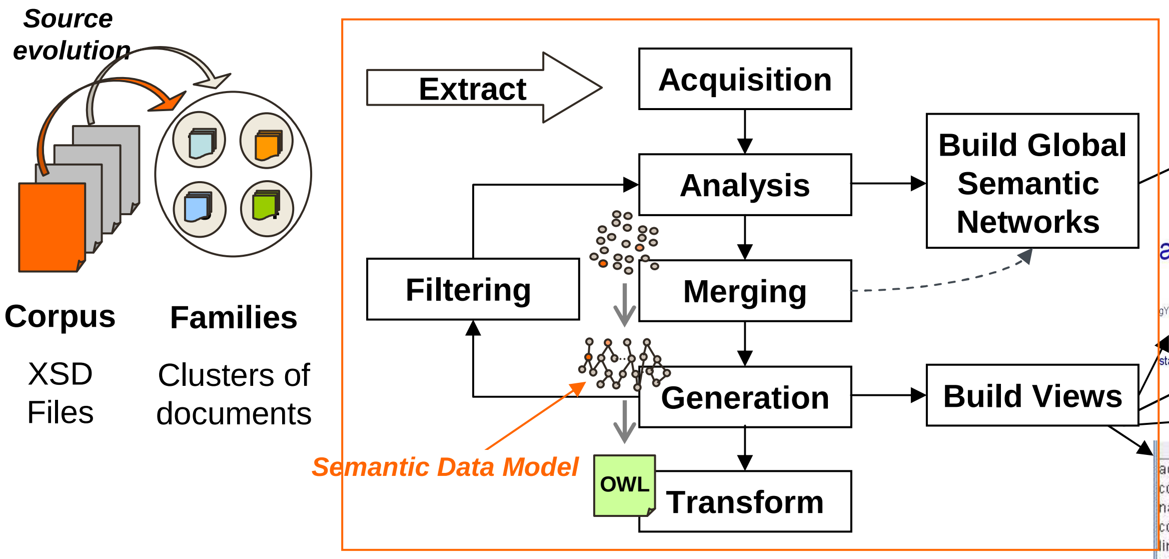


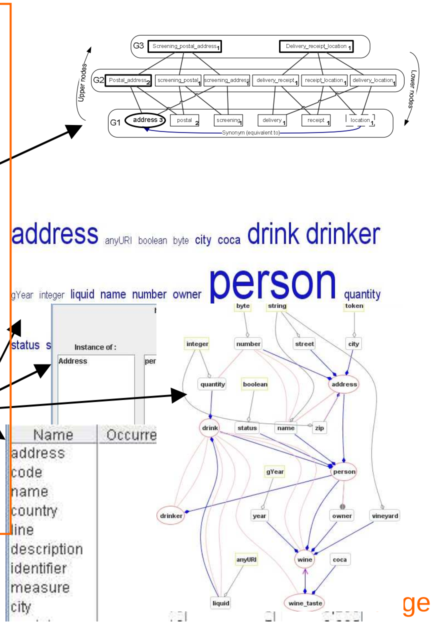
Janus

(the Roman god of gates and doors, beginnings and endings)

*

- Automatic tool for building ontologies from XSD Files
  - Implements XML Mining techniques (an adaptation of several techniques originating from the text mining and information retrieval/extraction fields, applied to XML files)
- The purpose are:
  - build as automatically as possible a system able to acquire and add knowledge *on the fly* from a corpus source (currently XSD is supported)
  - maintain machine centric collective memory to facilitate the discovery of concept similarities



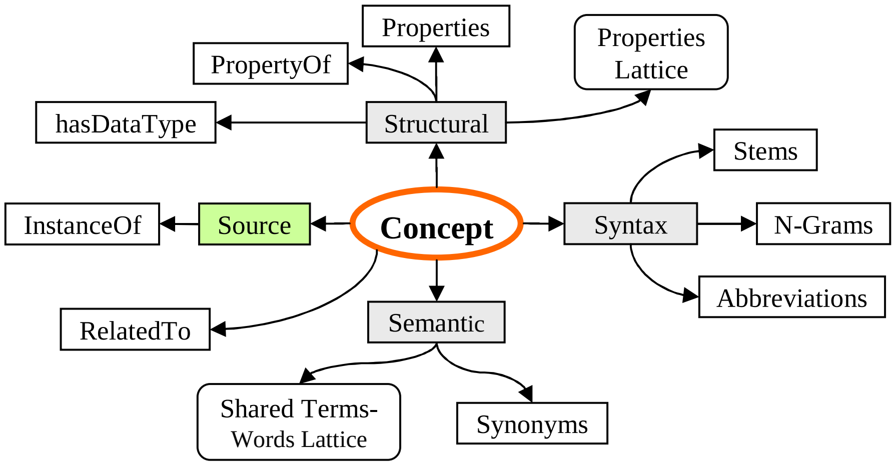


Name	Occurre
address	
code	
name	
country	
line	
description	
identifier	
measure	
city	

* www.microcarmuseum.com/tour/zundapp-janus.html

Janus : Semantic Data Model

- *Def 1*. Given a set of XSD files X as input source, we call **domain conceptualization *O*** *of* X, the set of concepts obtained by the application of a surjective mapping $m : X \rightarrow O$.
- *Def. 2*. A **concept** is the basic element of O and is defined as a quadruple $c = <L, Hc, Rc, I>$



- *Def. 3.* $c \in O$ *is a* ***class*** *if* $\exists P(c)=\{c_1, \ldots, c_m\}$, *where* $c_i \in O$ *and* $m > 1$. $C \subset O$ is the set of concepts classes
- *Def. 4.* $c \in O$ *is a* ***property*** *if* $\exists c_x \in C \mid c \in P(c_x)$ $P \subset O$ is the set of concepts properties
- *Def. 5.* $c \in O$ is a **data-type**, also called **printable** type, if $P(c)=\varnothing$

$m : X \rightarrow O$	
XSD Structure	**Mapping to *O***
xs:complexType	Concept class
xs:complexType with declared xs:simpleContent	Concept datatype
Element with attribute "ref" to xs:complexType	Concept class with *propertyOf* relationship
Named xs:element with attribute "type"	Concept class with *Is a* relationship
Named xs:element	Concept class
xs:simpleType	Concept datatype
Attributes of xs:element and xs:compleType	Concept properties
xs:extension et xs:restriction	Datatype property and *is a* relationship
xs:union	ComplexType properties
xs:any	Datatype property of the correspondent concept
xs:minOccurs, xs:maxOccurs	Respective cardinalities
xs:sequence, xsd:all	Concept properties
xs:choice	Disjointness concepts

Janus: Extraction

A Brief Introduction to XML Mining

- The surjective mapping $m : X \rightarrow O$ realizes the XML Mining operation. It also provides the following tasks:
  - Normalization. Extracted tag names may contain syntactic variation around the “core” concept, thus data are normalized in order to discover similarities around a "core" concept (e.g.: *PostalAddress* ⇔ *DeliveryLocation* ⇔ *Addr*)
    1. Checking composite words (e.g.: on-line)
    2. Remove identified useless-words (e.g.: CommonData for UnitOfMeasureCodeCommonData)
    3. Tokenization of tag labels considering the UCC convention, ‘_’ and ‘-‘ as separators (e.g.: <PersonIdentification_Type> = person + identification)
    4. Check for abbreviation (e.g.: Addr = Address, PO = Purchase Order)
    5. Remove stop-words (like “the”, “a”, “for”,…)
    6. Remove unknown words (dictionary based)
    7. Words Lemmatization (the canonical form of a word or set of word) and Stemming
    8. Synonym detection (dictionary based)
    9. Tag normalization (e.g.: *parse_resource_identifier* for *ParsedResourceIdentifier2_Type*)
  - Tag Frequency measure
    - TF calculated relatively to the frequency from extracted files and the number of family where the tag appears: $NormTagF(i,j) = w_i * TagF(I,j) / max(TagF(I,j))$

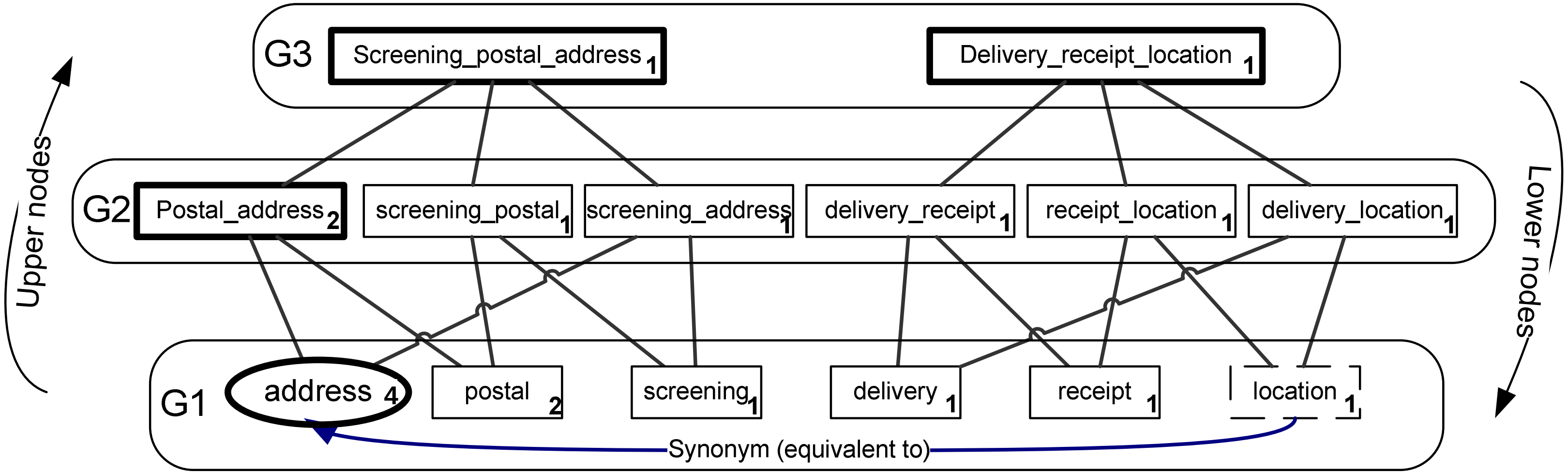
Janus: Semantic Network of Tags

Naming Affinity

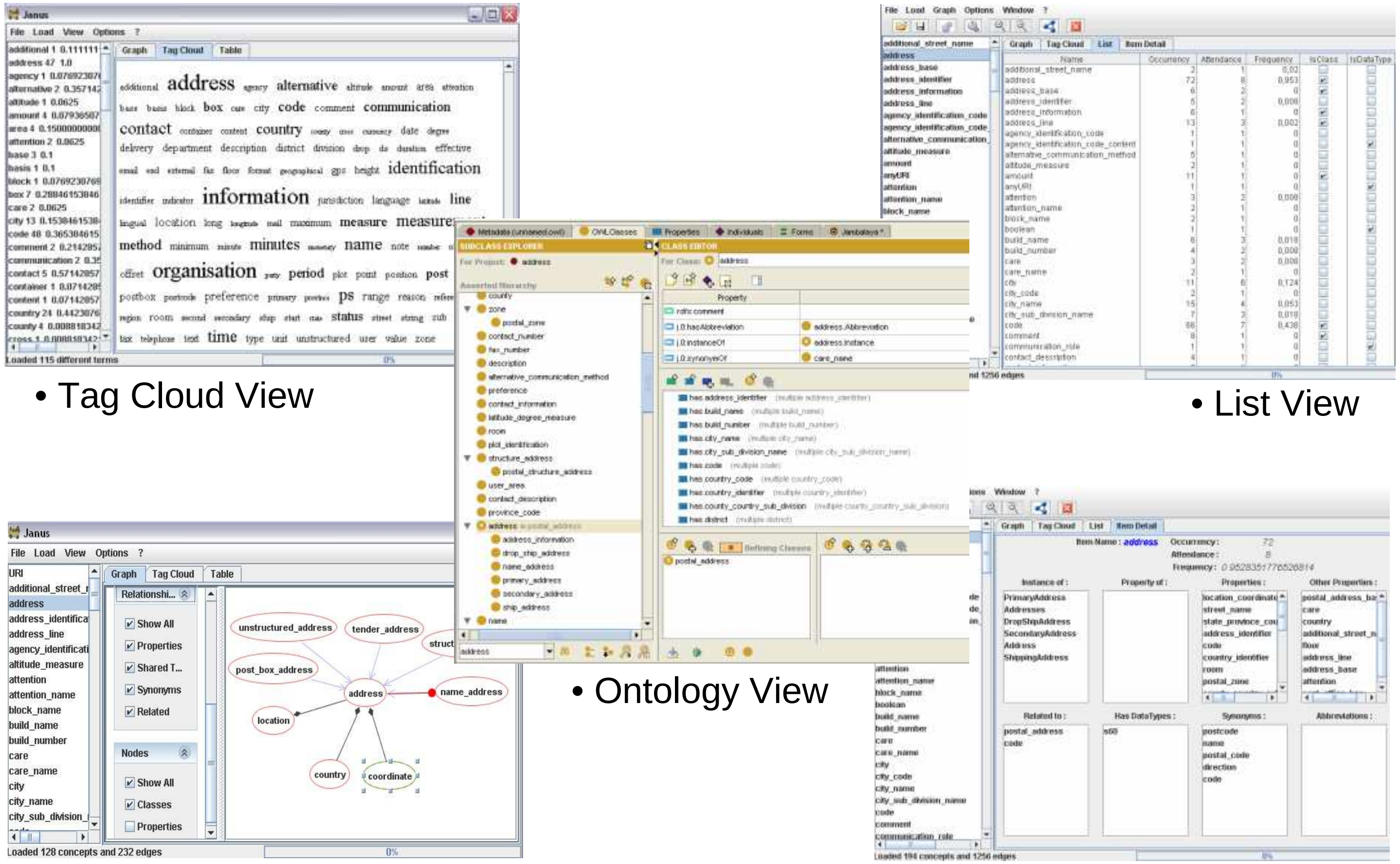
- Galois Lattice method and frequency-based strategy permit
  - To find the most important name for a concept carried by a set of tags at semantic level
  - To build a neighborhood of nodes to improve computational time when look for possible matchings

Ex.: considering the following tags:

- ***Address, PostalAddress, ScreeningPostalAddress, DeliveryReceiptLocation, Addr.***



Janus: Views and Ontology Generation


Janus
File Load View Options ?
Graph Tag Cloud Table
additional address agency alternative altitude amount area attention
base basis block box care city code comment communication
contact container content country county cross currency date degree
delivery department description district division drop dr duration effective
email end external fax floor format geographical gps height identification
identifier indicator information jurisdiction language latitude line
lingual location long longitude mail maximum measure measurement
method minimum minute minutes monetary name note number
offset organisation party period plot point position post
postbox postcode preference primary province ps range reason reference
region room second secondary ship start state status street string sub
tax telephone text time type unit unstructured user value zone
Loaded 115 different terms
URI
additional_street_name
address
address_identifier
address_line
agency_identification
altitude_measure
attention
attention_name
block_name
build_name
build_number
care
care_name
city
city_name
city_sub_division_name
Relationshi...
☑ Show All
☑ Properties
☑ Shared T...
☑ Synonyms
☑ Related
Nodes
☑ Show All
☑ Classes
□ Properties
unstructured_address
tender_address
post_box_address
address
name_address
location
country
coordinate
Loaded 128 concepts and 232 edges
SUBCLASS EXPLORER
CLASS EDITOR
For Class: address
rdfs:comment
hasAbbreviation
instanceOf
synonymOf
address.Abbreviation
address.Instance
care_name
Name Occurrency Attendance Frequency isClass isDataType
Item Name : address Occurrency : 72 Attendance : 8
Instance of : Property of : Properties : Other Properties :
PrimaryAddress
Addresses
DropShipAddress
SecondaryAddress
Address
ShippingAddress
Related to : Has DataTypes : Synonyms : Abbreviations :
postal_address
code
postcode
name
postal_code
direction
Loaded 194 concepts and 1256 edges

- Tag Cloud View
- List View
- Ontology View
- Graphical View
- Concept Detail View