Transforming XML Schema to OWL Using Patterns

Ivan Bedini, Christopher Matheus, Peter F. Patel-Schneider, Aidan Boran
Bell Labs Research

Benjamin Nguyen
University of Versailles St-Quentin & INRIA-Rocquencourt

IEEE-ICSC 2011

AGENDA

1. B2B Use Case
2. XML Semantics
3. Transformation Patterns and Rules
4. System Evaluation
5. Conclusion

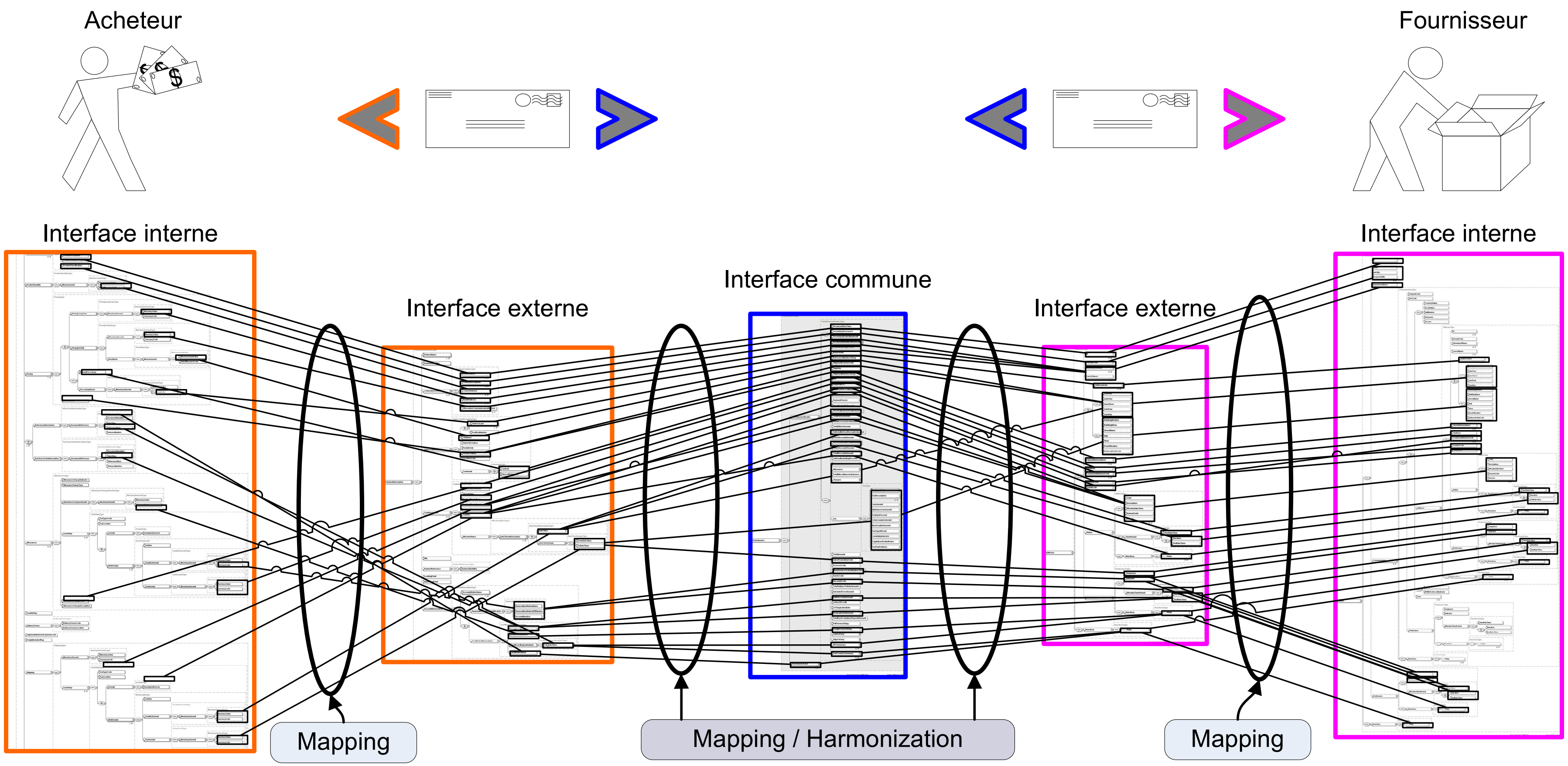
AGENDA

1. **B2B Use Case**
2. XML Semantics
3. Transformation Patterns and Rules
4. System Evaluation
5. Conclusion

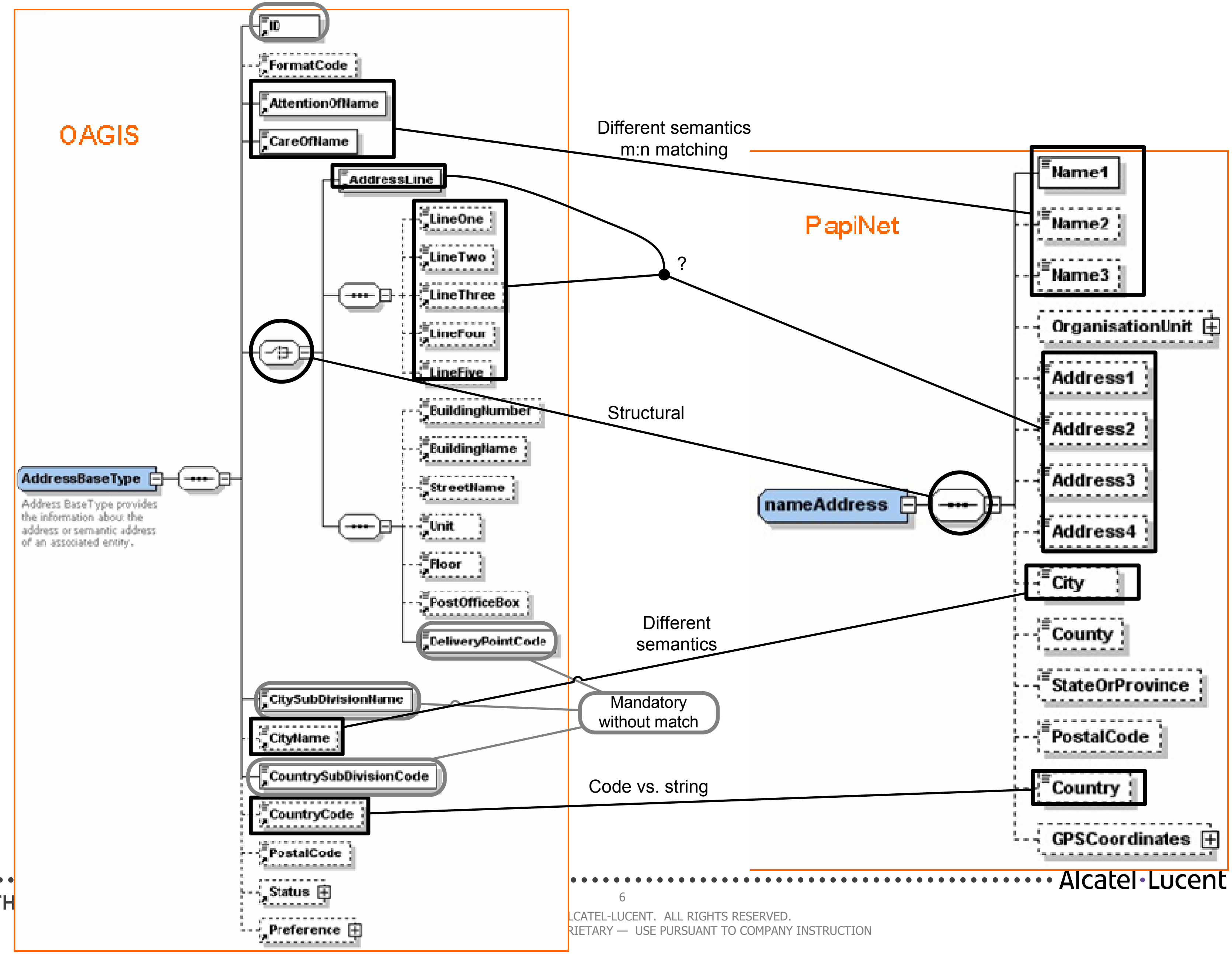
B2B Standards

- **40+ B2B standards** have been analyzed
- Most of them specify :
  - Data Dictionaries, Messages, WSDL, Business Processes, Code Lists and EDIFACT messages
- Formats :
  - EDIFACT, less and less provided
  - DTD, very few
  - Ontologies, mostly research work yet, but showing growing interest
  - JSON, no one
  - XML Schema (XS), **everyone**

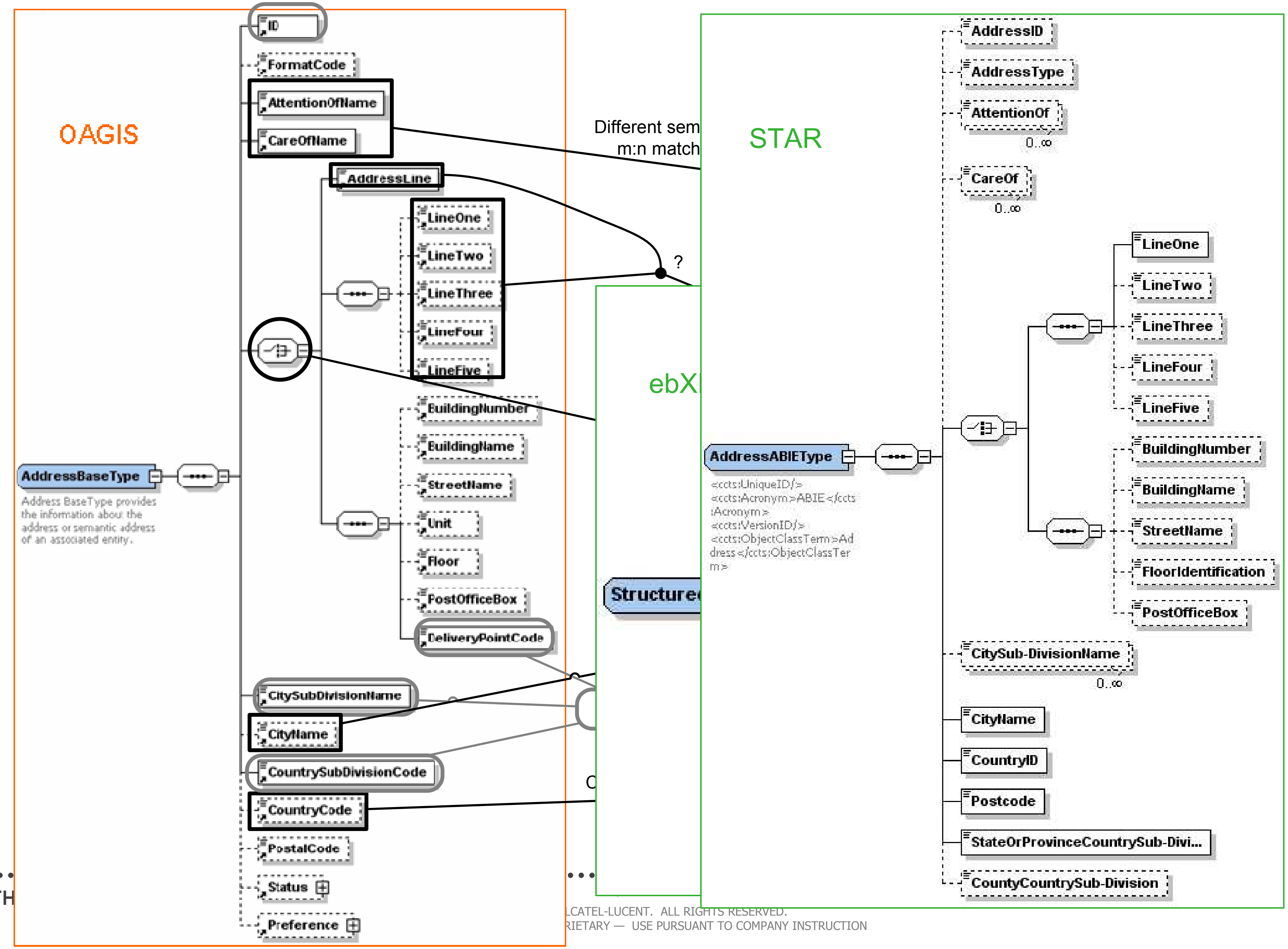
B2B Use Case Scenario



XML Documents and XML Schemas



XML Documents and XML Schemas



AGENDA

XML Semantics: Names’ Derivation for OWL Entities

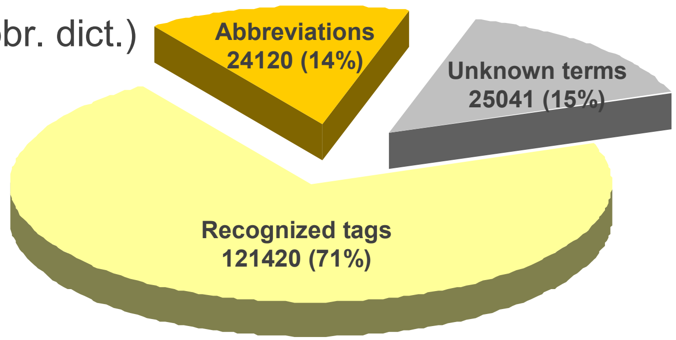
- Having proper entity names improves the transformation process
  - providing a meaningful ontology
  - improving the integration of multiple sources
  - simplifying the matching operations with other resources

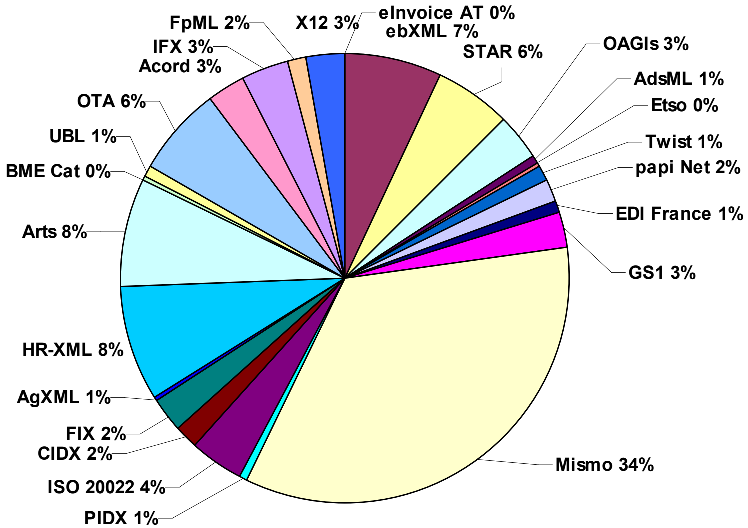
XML Semantics: Names' Derivation for OWL Entities

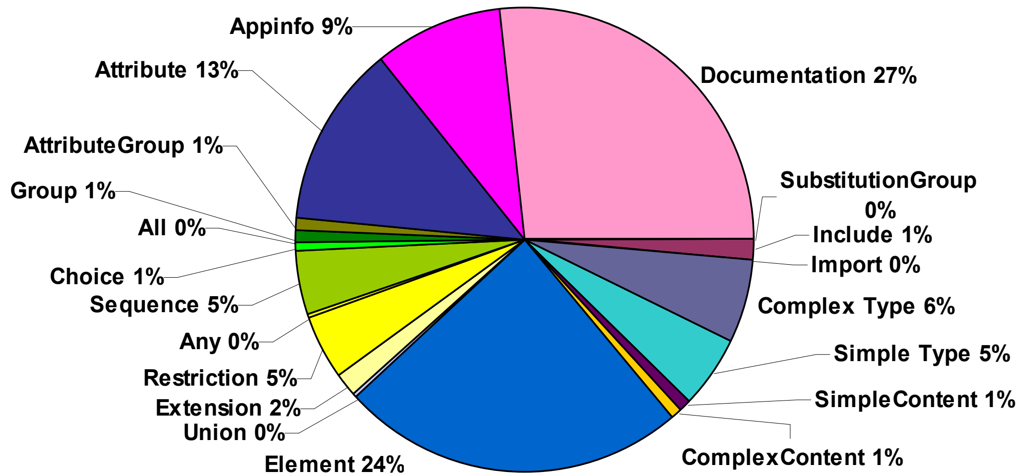
- XML Schema design have a lot of different practices on naming elements that not always are of direct understanding
  - Normal words using *Camel Case* convention, like *OfficeLocation*
  - Abbreviations, like *amt_ccy* (= *amount currency?*)
  - Acronyms, like PO (*Post Office* or *Purchase Order*?)
  - Compound words, like *cash-flow*, but sometime not separated, like *foodservice*
  - Misspelled words
  - Specific terms
  - Prefix or a suffix, like *_type, TYPE, _tp, _t, _T, type_*
  - Unrelated words with the meaning of the element, like *UnitOfMeasure**BBIECommonData***

XML Mining: Some Figures

- Statistics based on 25 B2B standards
  - 3432 XSD Files, +586.000 XML Components, +170.000 named
  - 145540 XML tags are composed by +3000 recognized terms
  - 85% of tags use real terms (based on WordNet and specific abbr. dict.)
  - 15% results to be difficult to get the semantic meaning,
- Improving the semantic recognition means:
  - more complex management of abbreviations
  - implementing NLP techniques
  - improving the external dictionaries capabilities







AGENDA

1. B2B Use Case
2. XML Semantics
3. **Transformation Patterns and Rules**
4. System Evaluation
5. Conclusion

XML Schema Components

- There are several XS components BUT!
  - 17 ways to declare elements (e.g. global/local element, references to a global element, a global/local element which defines a simple/complex type inline declaration, …)
  - 20 different ways to declare attributes.
- Components are divided in groups as follow:
  - **Named components (Nc)**: establishing the ontology entities (Classes (C), Datatypes (D))
    - simple type (St) definitions, complex type (Ct) definitions, attribute (A) declarations and element (E) declarations, attribute group (AG) definitions, model group (G) definitions, SubstitutionGroup (S).
  - **Relational XS constructs (R):** defining ontology properties (P) (OWL datatype and object properties)
    - sequence, all, choice, ref, simpleContent, complexContent, restriction, extension, SubstitutionGroup, union, any, min/max occurs.
  - **Helper components (H)**, which provide small parts of other components and are dependent on their context, mainly annotations.
- We define X = <Nc,R.Nc>, where Nc = U(A,E,G,AG,St,Ct,S)

Derivation of logical assertions from XML Schemas

- Ontologies and XML schemata serve very different purposes
  - Ontology languages are a means to specify domain theories based on logical representation
  - XML schemata are a means to provide syntactic and integrity constraints for information sources
- But have one main goal in common: both provide information about data vocabulary and "structure"
- The derivation methodology to permit the provision of not only basic information for a target ontology is a mix of rules and patterns
  - Capturing the most natural derivations
  - Interpret logical construct from some XML Schema

➔ We define the **domain conceptualization** *T* as the set of derived logical assertions from *X'*, the sub-set of *X*, having a corresponding element in *T*, through the surjection *m*.

$$m : X' \rightarrow T$$

$$X' = \{\forall x \in X \mid m(x,t),\ t \in T\} \subseteq X$$

$$T = \{U(C,D), P.T\}$$

Simple and Complex Types Patterns : Example

#	XS	OWL
1	<**simpleType** name="st_name">	:st_name rdf:type rdfs:Datatype .
2	<**simpleType** name="st_name"> <**union** memberTypes="st_name1 xsd:nativDataType ..."/> </simpleType>	:st_name owl:equivalentClass :st_name1 ; owl:equivalentClass xsd:nativeDataType; owl:equivalentClass ... .
3	<**complexType** name="ct_name">	:Ct_name rdf:type owl:Class .

```
1. <xs:simpleType name="DispositionType">
2. <xs:simpleType name="DispositionType">
     <xs:union memberTypes="
       CriminalDispositionTypes xs:string"/>
   </xs:simpleType>
3. <xs:complexType name="CoordinateType"/>
```

```
1. :disposition a rdfs:Datatype .
2. :disposition  owl:equivalentClass
      :criminalDisposition , xs:string .


3. :Coordinate a owl:Class .
```

Example 2

15	<**element** name="elt_name" type="**xsd:nativeDataType**">	:elt_name rdf:type rdfs:Datatype ; owl:subClassOf xsd:nativeDataType .
18	<**complexType** name="ct_name"> <**sequence**> <element **ref**="elt_name"/> (referring to a simple type)…	:has_elt_name_ct_name a rdfs:Datatype; rdfs:domain :ct_name ; rdfs:range :elt_name .
21	<**element** name ="name="Elt_name"> <**complexType**> …	:Elt_name rdf:type owl:Class .
22	<**element** name ="name="Elt_name"> <**complexType**> <**sequence**> <**element** name="elt_name" type="**xsd:nativeDataType**"/>	:has_elt_name a owl:DatatypeProperty ; rdfs:domain :Elt_name ; rdfs:range : xsd:nativeDataType.

15

```
<xs:element name="NationalNumber" type="xs:string"/>
<xs:element name="InternationalNumber">          (21)
  <xs:complexType>
    <xs:sequence>                                (22)
      <xs:element name="InternationalCode" type="xs:string"/>
      <xs:element ref="NationalNumber">          (18)
    <xs:sequence/>
  <xs:complexType/>
<xs:element/>
```

```
:nationalNUmber a rdfs:Datatype ;
  owl:subClassOf xs:string .

:InternationalNumber a owl:Class .

:has_international_code a owl:DatatypeProperty ;
  rdfs:domain :InternationalNumber ;
  rdfs:range :internationalCode .

:has_nationalNumber a owl:DatatypeProperty ;
  rdfs:domain :InternationalNumber ;
  rdfs:range xs:string .
```

Derived Types

#	XS	OWL
4	<simpleType name="st_name"> <restriction base="xsd:nativDataType"> <enumeration value="value1">...	:st_name owl:equivalentClass [rdf:type rdfs:Datatype; owl:oneOf ("value1"^^xsd:nativDataType ...)] .
5	<simpleType name="st_name"> <restriction base="basedDT"> <minInclusive value="value1"/> <maxInclusive value="value2"/> </restriction> </simpleType>	:st_name owl:equivalentClass [rdf:type rdfs:Datatype; owl:onDatatype :basedDT; owl:withRestrictions ([xsd:minInclusive "value1"^^:basedDT] [xsd:maxInclusive "value2"^^:basedDT])] .
6	<complexType name="ct_name"> <simpleContent> <extension base="xsd:nativeDataType">...	:Ct_name rdf:type owl:Class . :has_ct_name rdf:type owl:DatatypeProperty ; rdfs:domain :Ct_name ; rdfs:range xsd:nativeDataType .
7	<simpleType name="st_name"> <restriction base="basedDT"> <minExclusive value="value1"/> <maxExclusive value="value2"/> </restriction> </simpleType>	:st_name owl:equivalentClass [rdf:type rdfs:Datatype; owl:onDatatype :basedDT; owl:withRestrictions ([xsd:minExclusive "value1"^^:basedDT] [xsd:maxExclusive "value2"^^:basedDT])] .
8	<complexType name="ct_name"> <simpleContent> <extension base="st_name"> ...	:Ct_name rdf:type owl:Class . :has_ct_name rdf:type owl:DatatypeProperty ; rdfs:domain :Ct_name ; rdfs:range :st_name ; rdfs:subPropertyOf :has_st_name .
9	<complexType name="ct_name"> <simpleContent> <extension base="ct_name2"> (see #26, 27, 28)...	Same than #8 + :Ct_name rdfs:subClassOf :Ct_name2 .
10	<complexType name="ct_name"> <simpleContent> <restriction base="xsd:nativDataType"> (cf #4,5,6)...	:Ct_name rdf:type owl:Class . :ct_name_dt owl:equivalentClass [rdf:type rdfs:Datatype; (cf#4,5,6)] . :has_ct_name rdf:type owl:DatatypeProperty ; rdfs:domain :Ct_name ; rdfs:range :ct_name_dt .
11	<complexType name="ct_name"> <simpleContent> <restriction base="st_name"> ...	Same than #10 + :has_ct_name rdfs:subPropertyOf :has_st_name .
12	<complexType name="ct_name"> <simpleContent> <restriction base="ct_name2"> ...	Same than #11 + :Ct_name rdfs:subClassOf :Ct_name2 .
13	<complexType name="ct_name"> <complexContent> <extension base="ct_name2">...	:Ct_name rdf:type owl:Class ; rdfs:subClassOf :Ct_name2 .
14	<complexType name="ct_name"> <complexContent> <restriction base="ct_name2">...	:Ct_name rdf:type owl:Class ; rdfs:subClassOf :Ct_name2 .

Pattern Recognition Limitations

- OWL is more generally expressive than XML Schema, it is not possible to derive a direct transformation pattern for each OWL logical construct.
  - E.g. inverse, transitive and symmetric properties (*owl:differentFrom*, *owl:NegativePropertyAssertion* and *owl:PropertyChainAxiom*)

DL Expressivity	*Conversion*
F - Functional properties.	✓
E - Full existential qualification (Existential restrictions that have fillers other than owl:Thing).	✓
U - Concept union.	-
C - Complex concept negation.	-
S - An abbreviation for ALC with transitive roles.	~
H - Role hierarchy (subproperties - rdfs:subPropertyOf).	✓
R - Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness.	-
O - Nominals. (Enumerated classes of object value restrictions - owl:oneOf, owl:hasValue).	✓
I - Inverse properties.	-
N - Cardinality restrictions (owl:cardinality, owl:maxCardinality).	✓
Q - Qualified cardinality restrictions (available in OWL 2, cardinality restrictions that have fillers other than owl:Thing).	✓
(D) - Use of datatype properties, data values or data types.	✓

- Conversely, not all XML Schema integrity constraints are convertible into OWL
  - E.g. pattern and length constraints on data values (*<pattern value="[a-z] [a-z] [0-9]"/>, <xs:length value="8"/>,*)

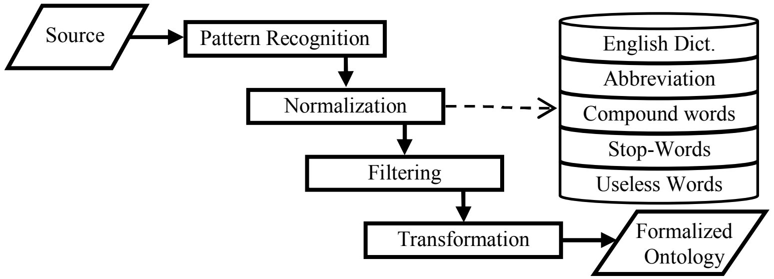
AGENDA

1. B2B Use Case
2. XML Semantics
3. Transformation Patterns and Rules
4. **System Evaluation**
5. Conclusion

System Implementation

- First step **parses** XSD sources and recognise selected **patterns** and stores candidate ontology entities in an organized internal data model based on multiple hash-tables.
- Second step **normalizes** extracted labels and produces real semantics, using:
  - WordNet version 3.0 using JWNL as English dictionary Other dictionaries are s
  - Specific lists of abbreviations, acronyms, stop-words, *compound words* and "*useless words"* (tailored for the specific domain but simply replaceable and adapteble)
- The following **filtering** step identifies more abbreviations eventually not detected previously and purges sources that do not produce a semantically well structured output,
- The last step simply **translates** the normalised internal data model to OWL

- Implementation
  - Java as programming language
  - SAX as XSD parser
  - OWL-API to generate OWL ontologies



Transformation Comparison

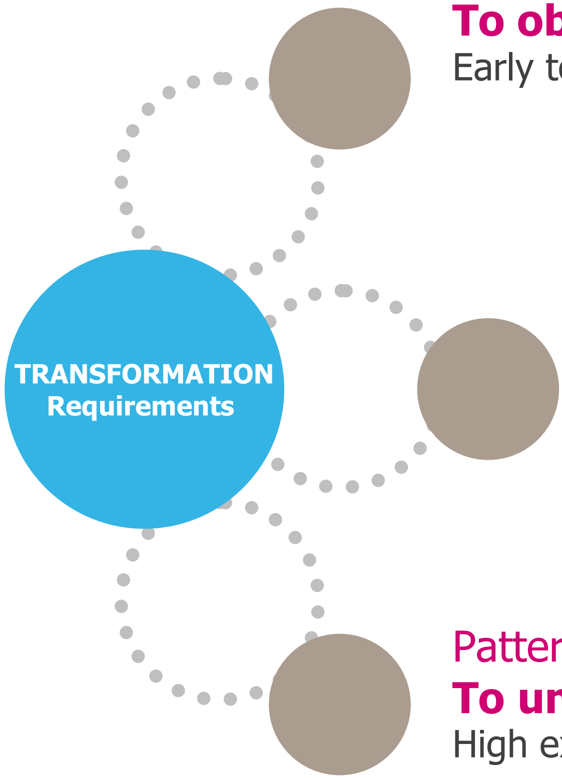
- Number of XSD constructs, appreciate the completeness of the map
- XML instances, produces OWL individuals
- Extensibility of the system
- Exception management, not a simple direct mapping
- Semantic normalisation
- Concept structures, to resolve hierarchical, properties and datatype relations
- Concept relations to get the richness of semantic relations
- OWL expressivity

	XML2OWL	**OWLMAP**	**LDM**	**Janus**
N. of XS construct	**8**	**9**	**18**	**19**
XML instances	✓	✓		
Extensible			✓	✓
Exception management	**limited**	**limited**	✓	✓
Semantic normalisation				✓
Concept structures	**limited**	✓	✓	✓
Concept relations	**limited**	✓	**limited**	✓
OWL expressivity	**ALUHN**	**tbd**	**tbd**	**ALHONQF(D)**

AGENDA

Alcatel·Lucent

Conclusion



Semantics

To obtain meaningful ontology

Early topic to be further investigated

Rules

To realize basic conversions

Highly adopted already

Patterns

To unveil logical assertions

High extensible contribution provided with this work

Thank you!

www.alcatel-lucent.com