Semantic Support for Electronic Business Document Interoperability

Asuman Dogac, Yalin Yarimagan, Yildiray Kabak

Middle East Technical University
and
Software Research, Development and Consultancy Ltd.
Ankara, Turkey

This work is supported by the European Commission through the ICT 213031 iSURF Project: http://www.iSURFProject.eu

The Motivation of this work…

- The European Commission's "Enterprise Interoperability Research Roadmap" foresees a "Interoperability Service Utility (ISU)"
  - "Interoperability as a utility-like capability needs to be supported by an enabling system of services for delivering basic interoperability to enterprises, independent of particular IT deployment"
  - http://cordis.europa.eu/ist/ict-ent-net/ei-roadmap_en.htm

- A very important component of "Interoperability Service Utility" is the interoperability of the business document instances exchanged through the service utility
- This work is being realized within the scope of the ICT 213031 iSURF Project
  - http://www.iSURFProject.eu

Talk Outline

- A Brief Overview of Electronic Business Document Standards
- UN/CEFACT Core Component Technical Specification
- Semantic Tools for Interoperability Support
  - Use of Ontologies for Semantic Annotation and Ontology Alignment
  - Document Translation
  - System Architecture and Operation
- Conclusions

Development of Electronic Business Document Interoperability Standards

- The development of electronic business document standards has been evolutionary based on:
  - The traditional EDI technology
  - Affected by the technological developments such as the Internet and XML
  - Affected by the interoperability needs of the current more dynamic eBusiness applications
- No document standard is sufficient for all purposes because the requirements significantly differ
  - Amongst businesses, industries and geo-political regions

Some Example Business Document Standards

- Vertical Standards
  - RosettaNet, CIDX, PIDX, OTA, HL7, …
- Horizontal Standards
  - OAGIS, GS1 eCom, xCBL, cXML, UN/CEFACT CCL, UBL, …
- A survey and analysis of electronic business document standards investigating:
  - The document design principles
  - The use of code lists
  - The use of XML namespaces
  - How the standards handle extensibility and customization
- is available at:
  - Kabak Y., Dogac A., “A Survey and Analysis of Electronic Business Document Standards”, Submitted to **ACM Computing Surveys**
    - http://www.srdc.metu.edu.tr/webpage/publications

UN/CEFACT Core Component Technical Specification (CCTS)

- The ultimate aim of business document interoperability is to
  - Exchange business data among partners without any prior agreements related to the document syntax and semantics
  - Hence support "Interoperability Service Utility (ISU)" at the content level
- Therefore, document standard need to adapt to different contexts, be extensible and customizable
- UN/CEFACT Core Component Technical Specification (CCTS) is an important landmark in this direction

Talk Outline

- A Brief Overview of Electronic Business Document Standards
- UN/CEFACT Core Component Technical Specification
- Semantic Tools for Interoperability Support
  - Use of Ontologies for Semantic Annotation and Ontology Alignment
  - Document Translation
  - System Architecture and Operation
- Conclusions

UN/CEFACT Core Component Technical Specification (CCTS)

- UN/CEFACT CCTS provides a methodology to identify a set of reusable building blocks, called Core Components to create electronic documents
- Core Components represent the common data elements of everyday business documents such as “Address”, “Amount”, or “Line Item”
- These reusable building blocks are then assembled into business documents such as “Order” or “Invoice” by using the CCTS methodology
- UN/CEFACT CCTS Core Components are syntax independent

UN/CEFACT Core Component Technical Specification (CCTS)

- Core components are defined to be context-independent so that they can later be restricted to different contexts:
  - Business Process Context
  - Product Classication Context
  - Industry Classication Context
  - Geopolitical Context
  - Business Process Role Context
  - Supporting Role Context
  - System Capabilities Context
  - Official Constraints Context

Main Features of CCTS Approach

- Business document schemas are composed of several basic and aggregate components
- Aggregate components themselves are collections of other basic and aggregate components in a recursive manner
- Standard components are modified in response to contexual needs
- When a document schema needs to be customized for a context, users need to discover or provide component versions applicable to that particular context

Why CCTS is important?

- This concept of defining context-free reusable building blocks, which are available from a single common repository, is an important innovation:
  - The incompatibility in electronic documents is incremental rather than wholesale
  - The users are expected to model their business documents by using the existing core components and by restricting them to their context with well defined rules
  - Dynamic creation of interoperable documents becomes possible because if users cannot find proper components to model their documents, they can create and publish new core components
  - The horizontal interoperability among different industries is greatly facilitated by using a single common repository and by customizing the components to different industry contexts

Some of the UN/CEFACT CCTS based Business Document Standards

- UN/CEFACT Core Components Library (**CCL**) 07A
  - 96 ACC, 212 ASCC, 636 BCC
  - 184 ABIE, 337 ASBIE, 1011 BBIE
  - 35 Datatypes
- Universal Business Language (**UBL**) 2.0
- Open Applications Group Integration Specification (**OAGIS**) 9.0
- Global Standards One (**GS1**) XML
- **All standards implement CCTS differently!**

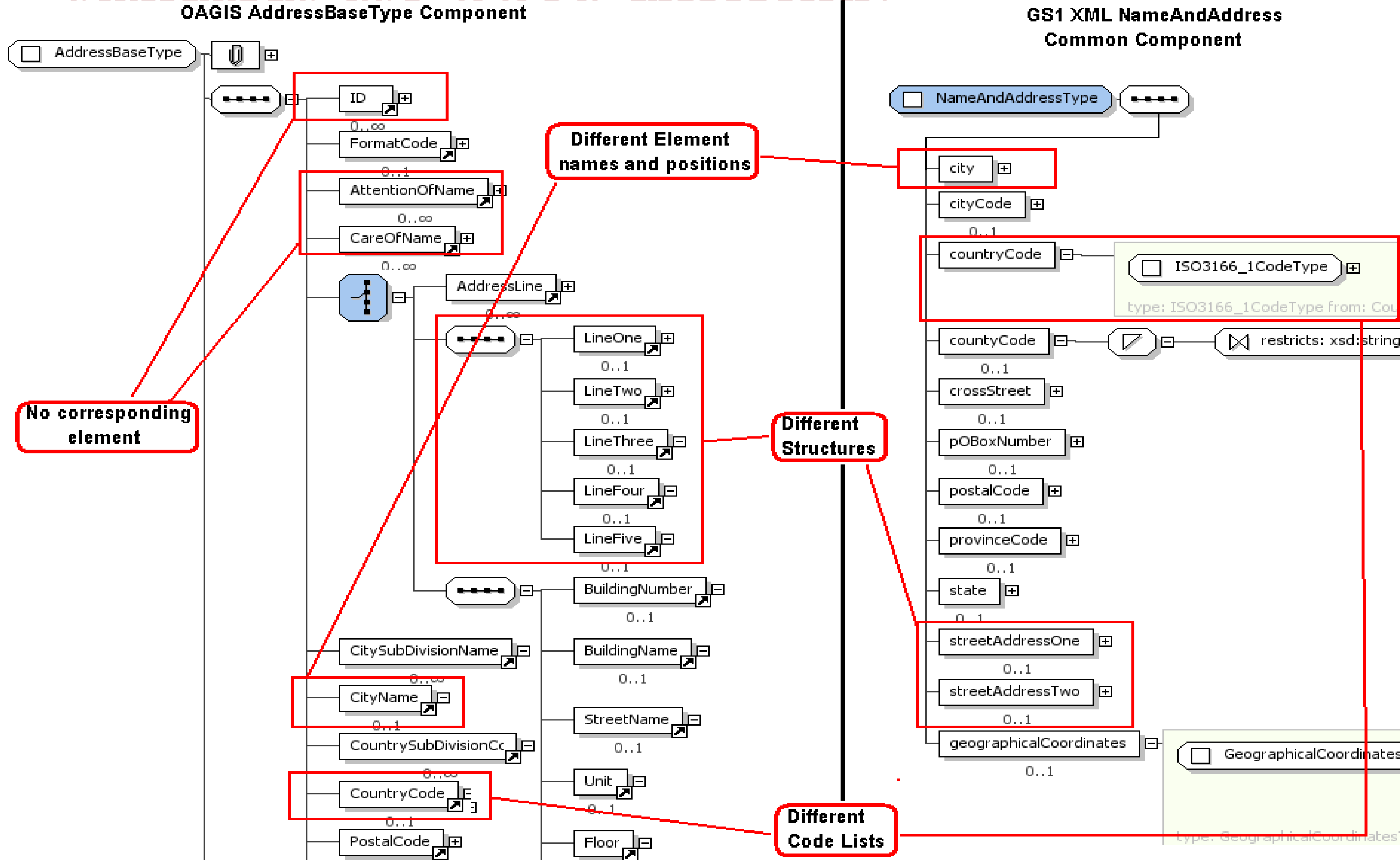
UN/CEFACT Core Components Library (CCL) 07A

	A	B	C	D	E	F	G	H	I	J	K
1	Action	Unique ID	Dictionary Entry Name (DEN)	ACC/ BCC/ ASCC	Definition	Library Note	Object Class Term Qualifier(s)	Object Class Term	Property Term Qualifier(s)	Property Term	Representation Term
2											
3				ACC	Aggregate Core Component						
4				BCC	Basic Core Component contained within the ACC						
5				ASCC	Associated (Aggregate) Core Component, associated with the ACC						
6				ABIE	Aggregate Business Information Entity						
7				BBIE	Basic Business Information Entity contained within the ABIE						
8				ASBIE	Associated (Aggregate) Business Information Entity, associated with the ABIE						
9		UN00001267	Accounting Account. Details	ACC	A specific account for recording debits and credits to general accounting, cost accounting or budget accounting			Accounting Account			
10		UN00001268	Accounting Account. Identification. Identifier	BCC	The unique identifier for this accounting account.			Accounting Account		Identification	Identifier
11		UN00001269	Accounting Account. Set Trigger. Code	BCC	A code specifying a set trigger for the accounting account to be used in response to a specific event or set of events.			Accounting Account		Set Trigger	Code
12		UN00001270	Accounting Account. Type. Code	BCC	The code specifying the type of accounting account such as general(main), secondary, cost accounting, budget account.			Accounting Account		Type	Code
13		UN00001271	Accounting Account. Amount Type. Code	BCC	The code specifying the amount type for a specific accounting account.			Accounting Account		Amount Type	Code
14		UN00000010	Address. Details	ACC	The location at which a particular organization or person may be found or reached.			Address			
15		UN00000011	Address. Identification. Identifier	BCC	A unique identifier for this address.			Address		Identification	Identifier
16		UN00000012	Address. Format. Code	BCC	A code specifying the format of this address.			Address		Format	Code
17		UN00000014	Address. Postcode. Code	BCC	A code specifying the postcode of the address.			Address		Postcode	Code
18		UN00000032	Address. Post Office Box. Text	BCC	The unique identifier, expressed as text, of a container commonly referred to as a box, in a post office or other postal service location, assigned to a person or organization, where postal items may be kept for this address.			Address		Post Office Box	Text
19		UN00000019	Address. Block Name. Text	BCC	The block name, expressed as text, for an area surrounded by streets and usually containing several buildings for this address.			Address		Block Name	Text
20		UN00000020	Address. Building Number. Text	BCC	The number, expressed as text, of a building or house on a street at this address.			Address		Building Number	Text
21		UN00000021	Address. Building Name. Text	BCC	The name, expressed as text, of a building, a house or other structure on a street at this address.			Address		Building Name	Text
22		UN00000023	Address. Room Identification. Text	BCC	The identification, expressed as text, of a room, suite, office or apartment as part of an address.			Address		Room Identification	Text
23		UN00000022	Address. Department Name. Text	BCC	A name, expressed as text, of a department within this address.			Address		Department Name	Text
		UN00000024	Address. Floor Identification. Text	BCC	The identification by name or number, expressed as			Address		Floor	Text

OASIS Universal Business Language (UBL) 2.0

- The first implementation of UN/CEFACT CCTS in XML
- 31 Horizontal Business Document Schemas
  - Invoice, Order, Dispatch Advice,…
- Schemas for common reusable entities
  - Amount, Payment, Item, …

The Problem continues: All CCTS based standards use CCTS differently



How to provide interoperability among electronic business document standards?

- Harmonization:
  - The International Electrotechnical Commission (IEC),
  - The International Organization for Standardization (ISO),
  - The International Telecommunication Union (ITU) and,
  - The United Nations Economic Commission for Europe (UNECE)

  signed a “Memorandum of Understanding” to specify a framework of cooperation

- Up to now, OAGIS 9.0 and UBL 2.0 have achieved a level of harmonization: they are based on the same UN/CEFACT Unqualified Datatypes and Core Component Types

- However, the harmonization needs to be extended to the upper level artifacts

- An alternative: Providing semantic tool support for the interoperability of electronic business documents

Providing semantic support for the interoperability of CCTS based electronic business documents

- Within the scope of the iSURF Project, we developed tools:
  - To provide machine processable semantic representations of context domains
  - To utilize these semantics for automating tasks for the discovery, reuse and customization of components and document schemas
  - To provide a semantics based translation mechanism for the interoperability of schemas customized by independent parties

Talk Outline

- A Brief Overview of Electronic Business Document Standards
- UN/CEFACT Core Component Technical Specification
- Semantic Tools for Interoperability Support
  - Use of Ontologies for Semantic Annotation and Ontology Alignment
  - Document Translation
  - System Architecture and Operation
- Conclusions

The Motivation: Context Categories

- Eight categories has been defined for the business context
- Specific code lists and classification schemas are suggested for each category:
  - Code lists and classification taxonomies provide context values
  - There are other relevant classifications in use today and there may be others in future
- Quoting from an email in the Ontolog Forum by Duane Nickull:
  - "*Even when the CCTS group decided to limit their context qualifier set to only 8 context aspects, they still had an almost infinite explosion of context. If you took 8 singular contexts and had only 300 enumerated values for each one, the number is so large no one group could ever possibly list all the combinations in a lifetime without computers*"

Context Ontologies

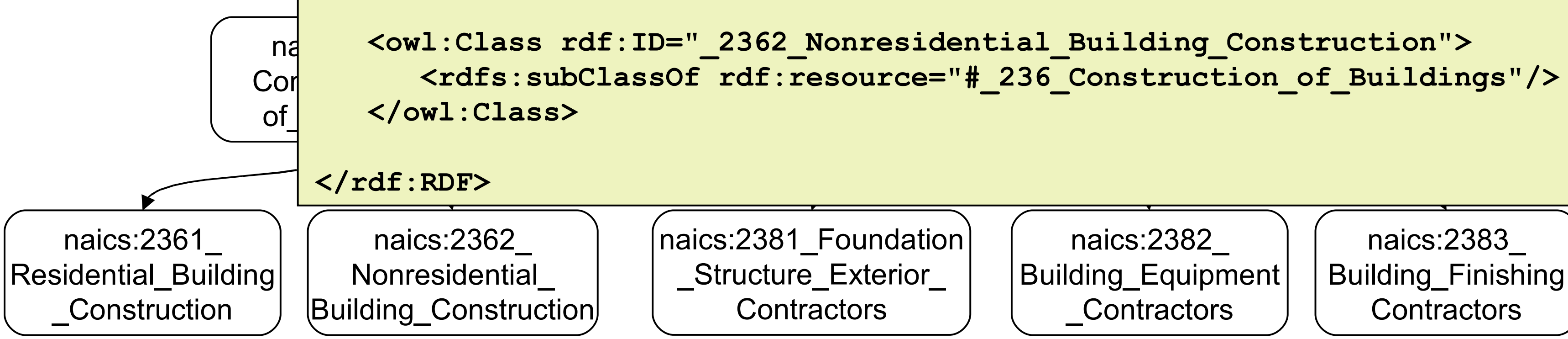
- We developed Web Ontology Language (OWL) ontologies to represent taxonomy of these classifications:
  - They become machine processable
  - It becomes possible to formally specify relationships between different classifications
  - Specified relationships are interpreted by reasoners to compute additional relationships

Context Ontologies

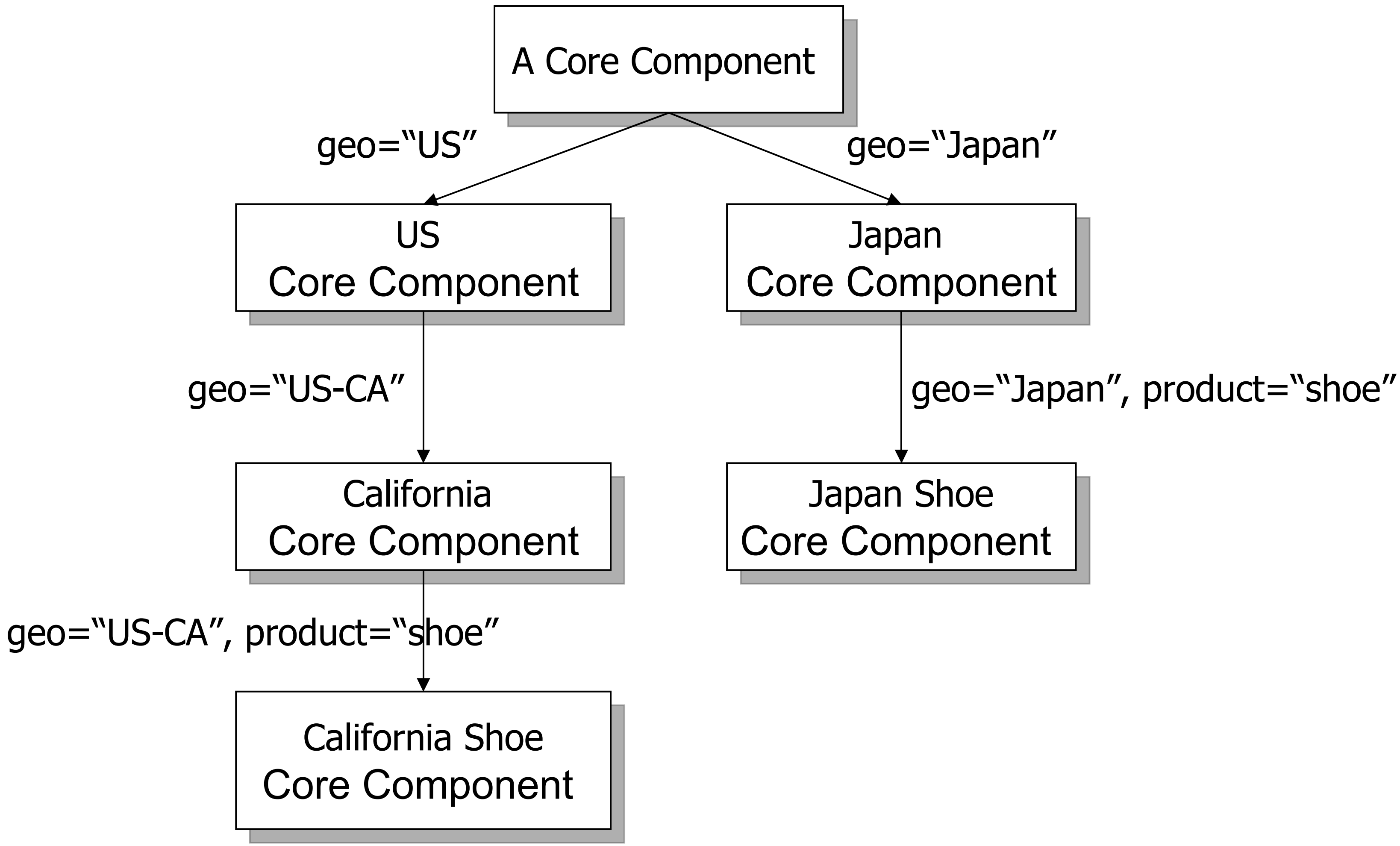
North American Indusrty Classification System (NAICS)

23	Construction
236	Const
2361	Resid
2362	Nonre
238	Speci
2381	Found
2382	Buildi
2383	Buildi

```
<?xml version="1.0"?>
<rdf:RDF
   <owl:Ontology rdf:about="NAICS Ontology"/>

   <owl:Class rdf:ID="_23_Construction" />

   <owl:Class rdf:ID="_236_Construction_of_Buildings">
      <rdfs:subClassOf rdf:resource="#_23_Construction" />
   </owl:Class>

   <owl:Class rdf:ID="_2361_Residential_Building_Construction">
      <rdfs:subClassOf rdf:resource="#_236_Construction_of_Buildings"/>
   </owl:Class>

   <owl:Class rdf:ID="_2362_Nonresidential_Building_Construction">
      <rdfs:subClassOf rdf:resource="#_236_Construction_of_Buildings"/>
   </owl:Class>

</rdf:RDF>
```

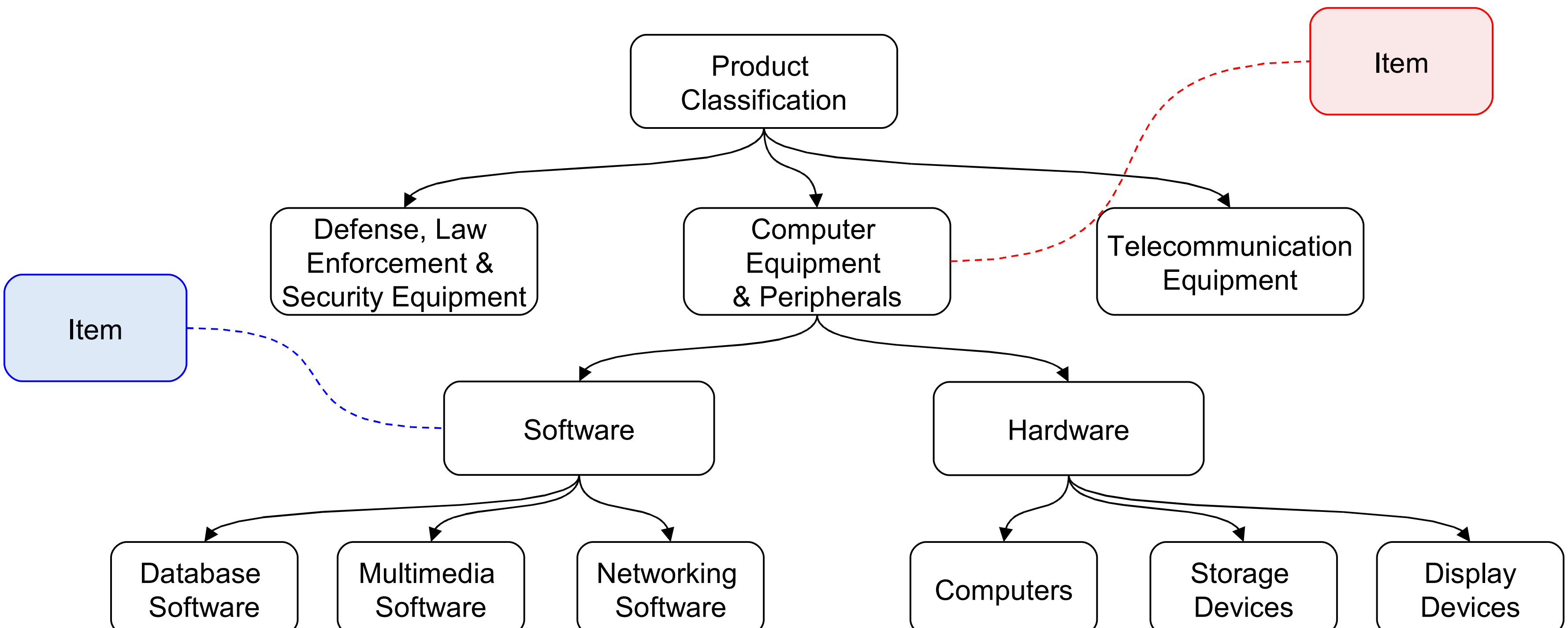


Context Based Customization



Influence of Custom Components

- Custom components are applicable for the context hierarchy they are defined for

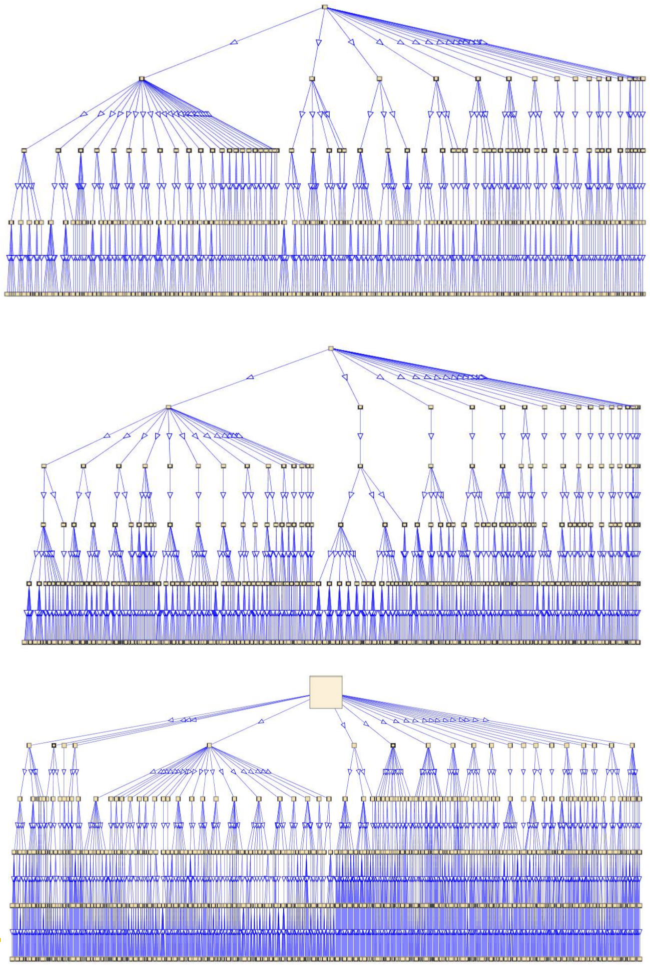


Context Ontologies

- We developed a tool to convert classifications to context ontologies in OWL representation:
  - Geopolitical context
    - M49, ISO-3166
  - Industrial Classification context
    - NAICS, NACE, ISIC
  - Product Classification context
    - CPC, UNSPSC

- <u>These context ontology classes are then used to annotate customized document components</u>

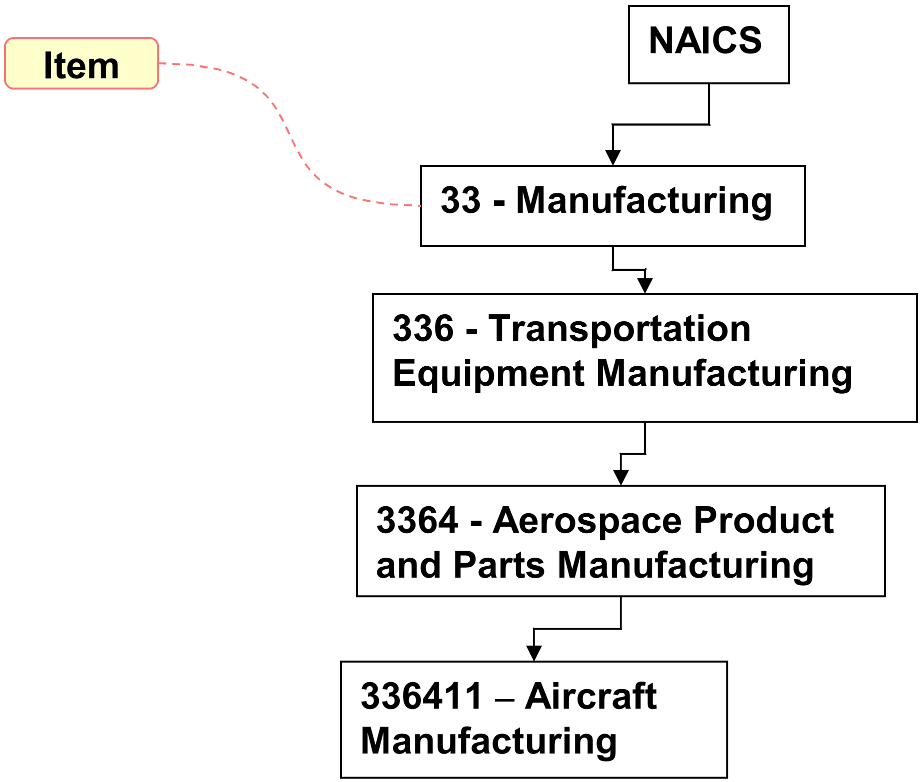
- Note: This is in addition to defining element values through code lists



Talk Outline

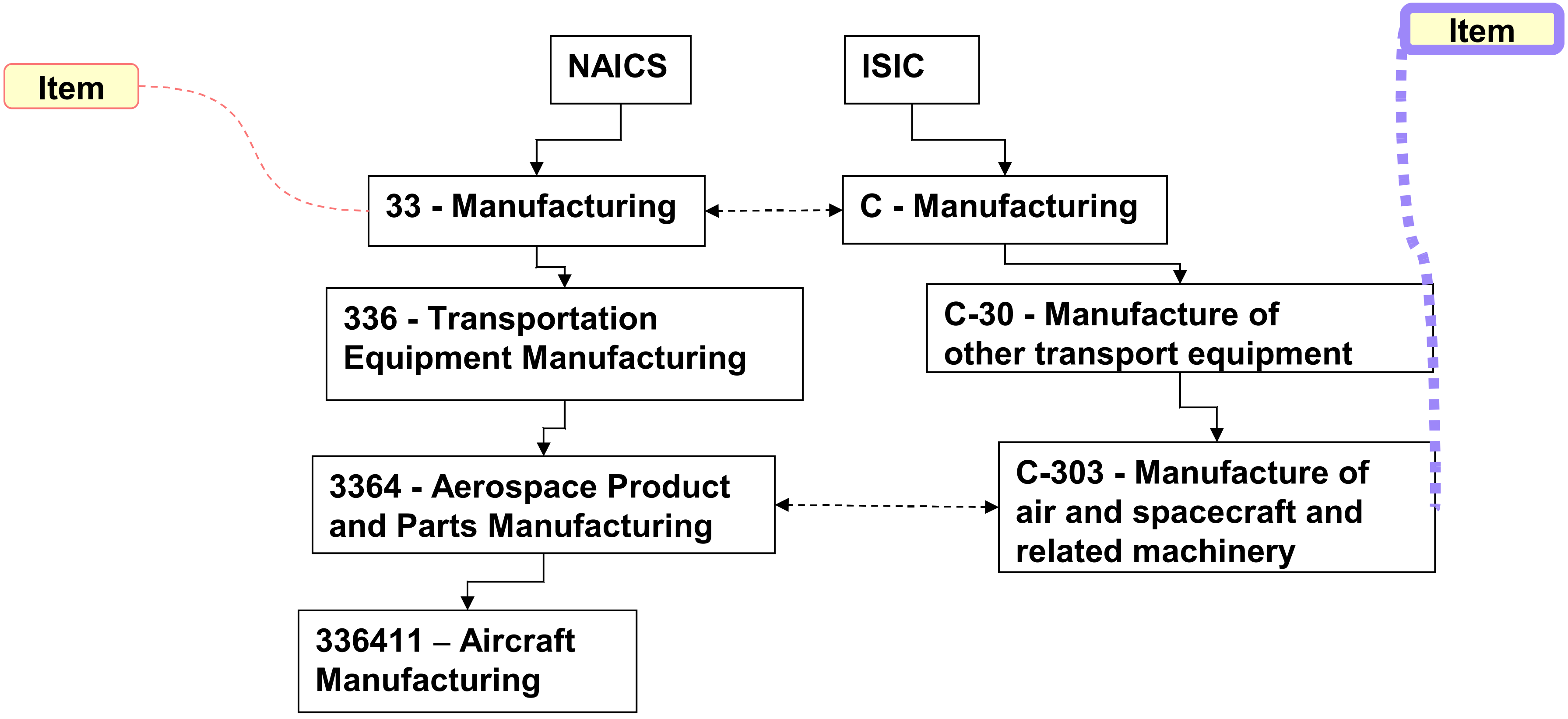
- A Brief Overview of Electronic Business Document Standards
- UN/CEFACT Core Component Technical Specification
- Semantic Tools for Interoperability Support
  - Use of Ontologies for Semantic Annotation and Ontology Alignment
  - Document Translation
  - System Architecture and Operation
- Conclusions

Annotating Components with Context Ontologies

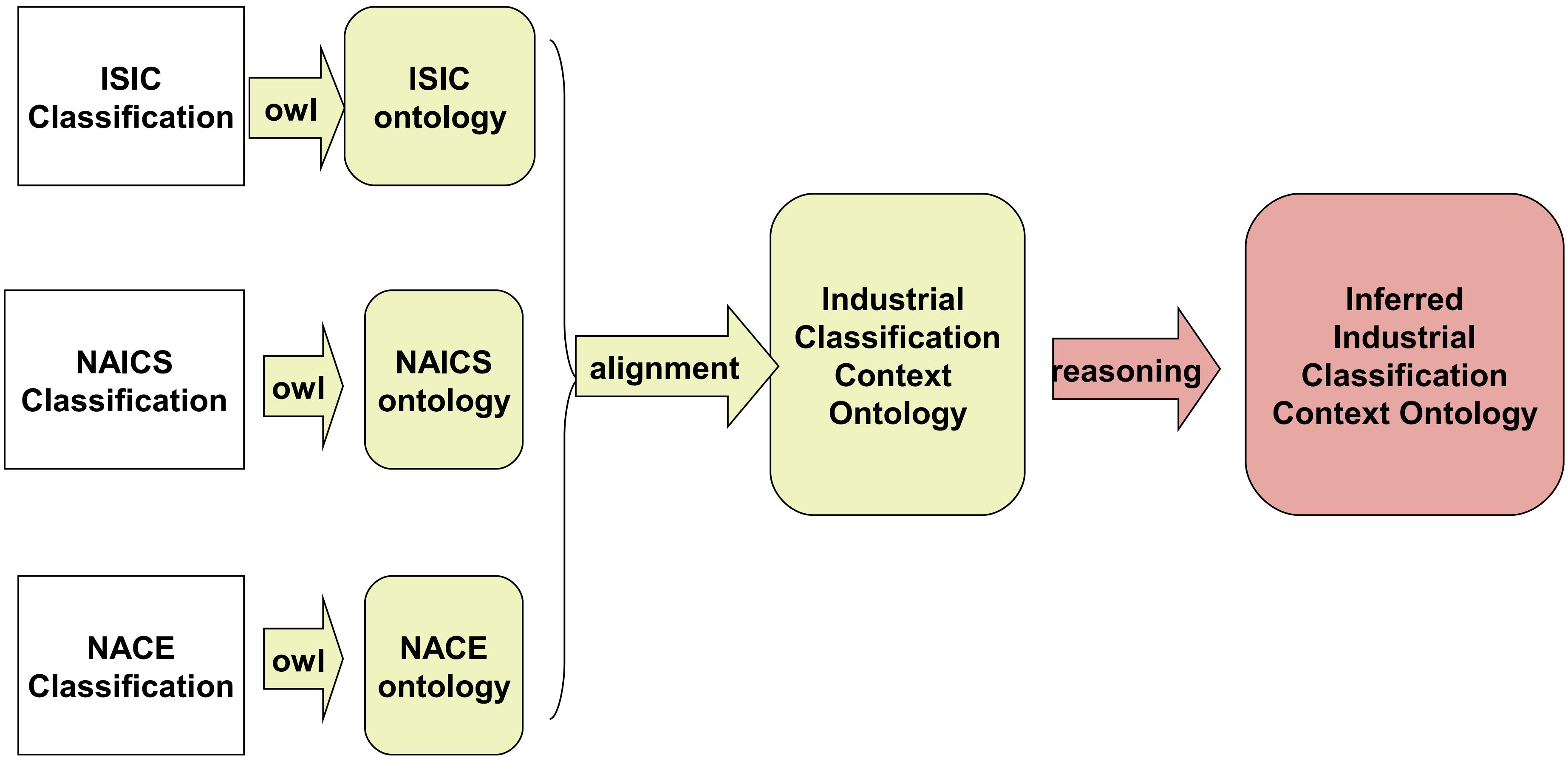


When a component "item" is defined for the "Manufacturing" context, it becomes applicable to all subclasses in the context ontology

Influence of Aligned Ontologies on Component Discovery and Reuse



Generating Context Ontologies



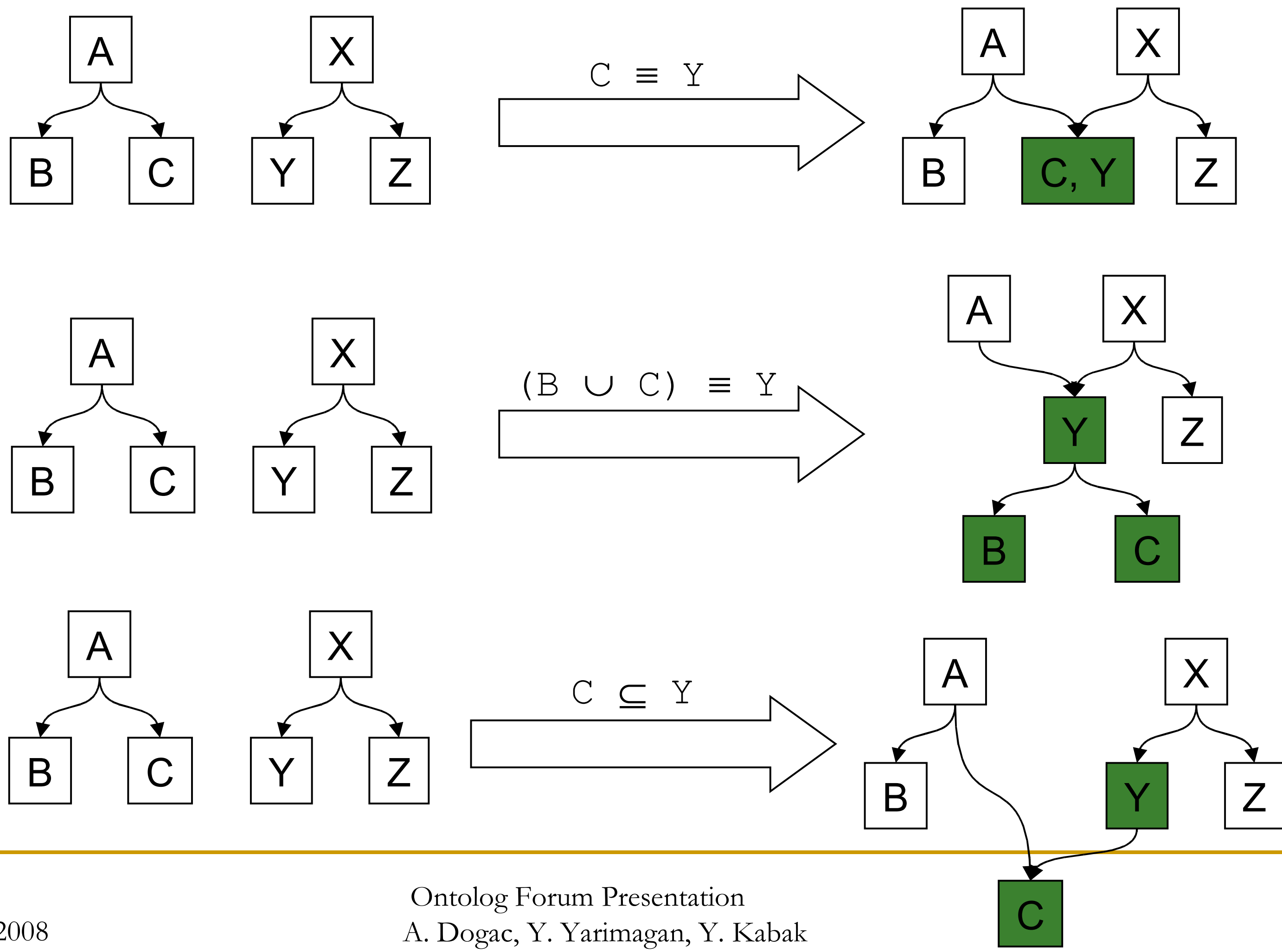
Aligning Context Ontologies

- A joint ontology is generated for each context category
  - Imports all ontologies relevant to that particular category
  - Allows additional ontologies to be added without effecting existing ones
  - Allows specification of correspondences between different ontologies
- Ontology alignment is to be assumed by domain experts and standard issuing bodies
- Our work focuses on how such correspondences can be exploited once they are specified

Aligning Context Ontologies

- Any OWL construct can be utilized including but not limited to:
  - Equivalence `(A ≡ B)`
    - `NACE:45-Construction, NAICS:23-Construction`
  - Composition `(A ≡ B ∪ C)`
    - `NAICS:11-Agriculture, Forestry, Fishing and Hunting, ISIC:A-Agriculture, Hunting and Forestry, ISIC:B-Fishing`
  - Subsumption `(A ⊆ B)`
    - `NACE:CA-Mining and Quarrying of Energy Producing Materials, NAICS:211-Oil and Gas Extraction`

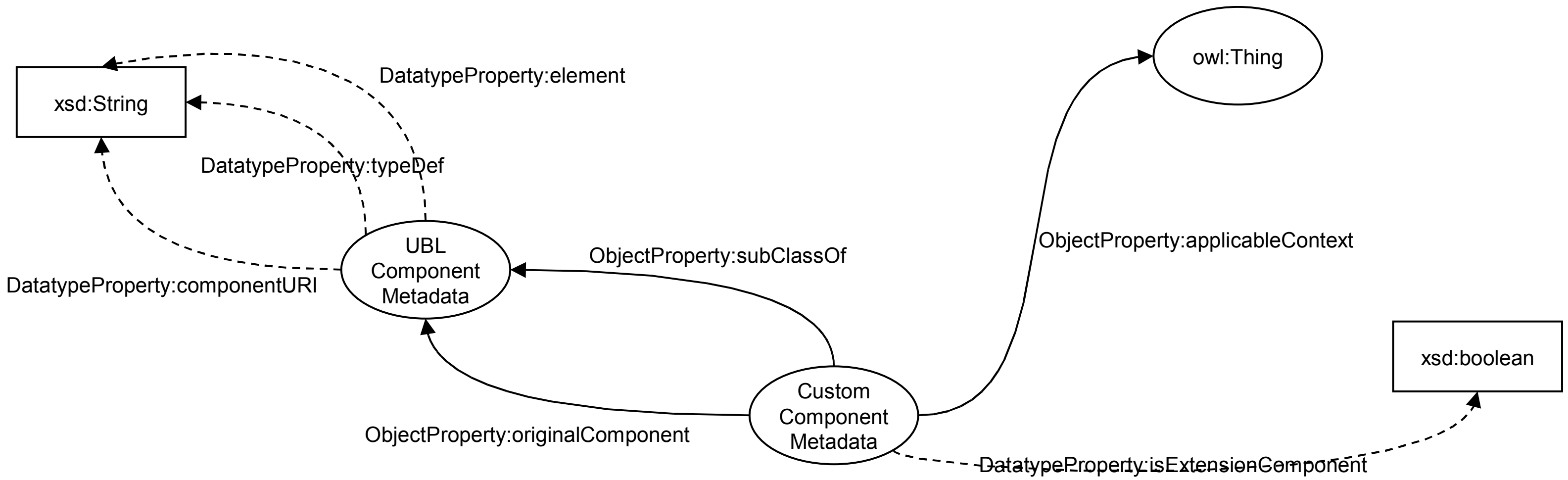
Ontology Alignment Operations



How to Annotate Components with Context Ontology: Component Metadata

- When a component is customized for a context, its metadata is created:
  - To express the standard component it is derived from, and
  - The context it is applicable to by specifying references to classes from ontologies
- When a custom version of a component is required for a specific context:
  - Component metadata is queried to gather applicable versions with the help of inferred context ontologies
- When a document schema needs to be customized for a specific context, component metadata is queried
  - To gather custom versions of components included in that schema and
  - Those versions are used to replace the original components in the customized document schema

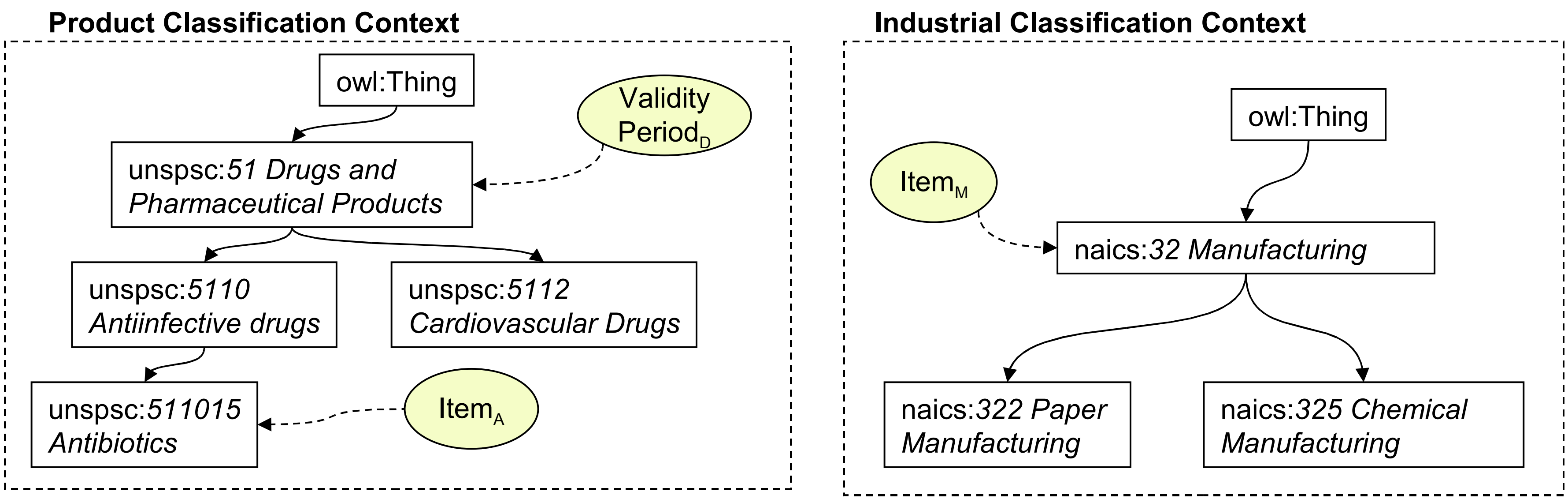
Component Metadata




```
<UBLComponentMetadata rdf:ID="cac_Item">
    <element rdf:datatype="string">urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2:Item</>
    <typeDef rdf:datatype="string">urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2:ItemType</>
    <componentURI rdf:datatype="string">http://www.srdc.metu.edu.tr/ublschema/common/UBL-CommonAggregateComponents-2.0.xsd</>
</UBLComponentMetadata>
```

```
<CustomComponentMetadata rdf:ID="Item-industry_naics_23_cnstrctn">
  <element rdf:datatype="string">srdc:industry:naics:_23_cnstrctn:ubl:Item</>
  <typeDef rdf:datatype="string">srdc:industry:naics:_23_cnstrctn:ubl:ItemType</>
  <componentURI rdf:datatype="string">http://srdc.metu.edu.tr/customSchemaRepository/industry_naics__23_cnstrctn.xsd</>
  <applicableContext rdf:resource="string">http://srdc.metu.edu.tr/contextOntology/naics.owl#_23_Construction</>
  <isExtensionComponent rdf:datatype="boolean">false</>
  <originalComponent rdf:resource=http://srdc.metu.edu.tr/componentRepository/ublInstances.owl#cac_Item</>
</CustomComponentMetadata>
```

Component Discovery Service



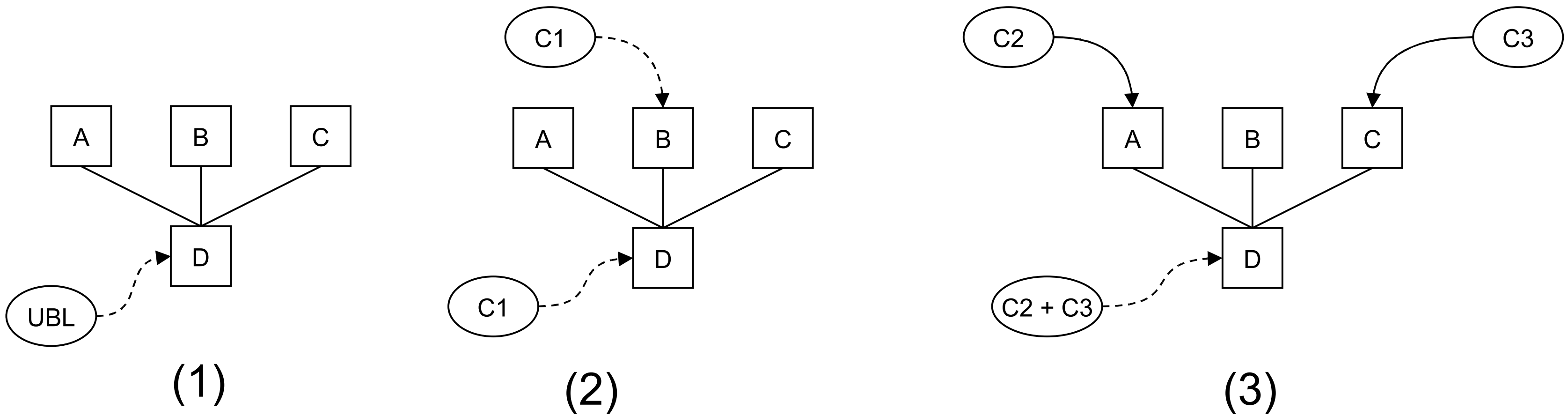
Item for *Antibiotics* context? **Item_A**

Item for *Cardiovascular drugs* context? **Item_{UBL}**

Validity Period for *Antibiotics* context? **ValidityPeriod_D**

Item for *Antibiotics Manufacturing* context? **Item_{A+M}**

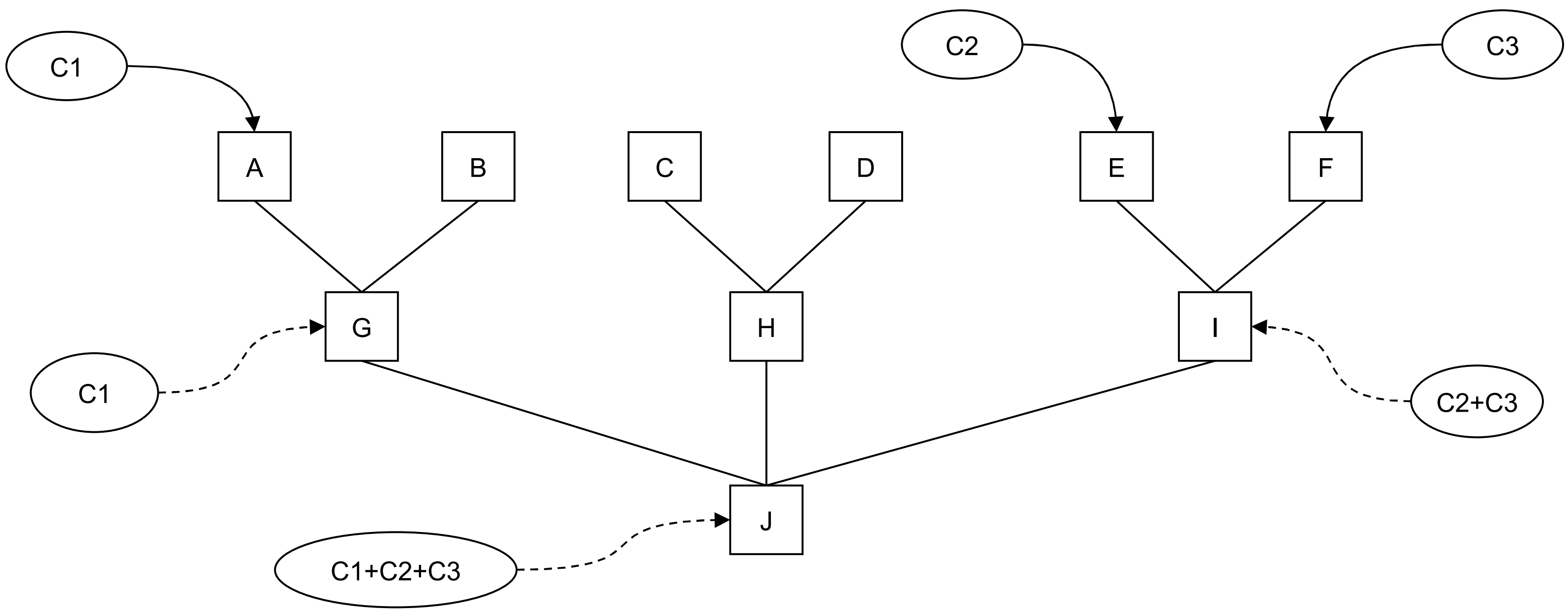
Component Discovery and Merging



1. If there are no customized components in the parent classes, the original standard component is used
2. If there is a customized component applicable to a parent context, for example, for class B, say "C1", this version is applicable to context class D
3. If there are customized components applicable to multiple parent context classes, for example, "C2" for class "A" and "C3" for class "C", the context applicable to class "D", is generated by merging the components "C2" and "C3"

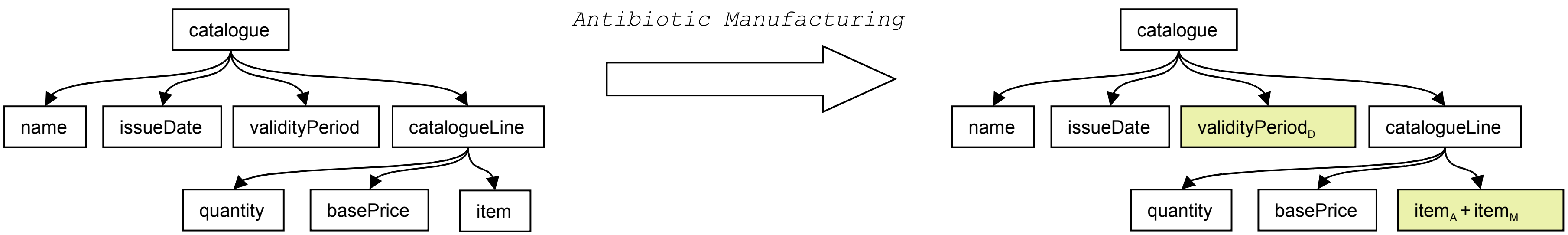
Component Discovery and Merging

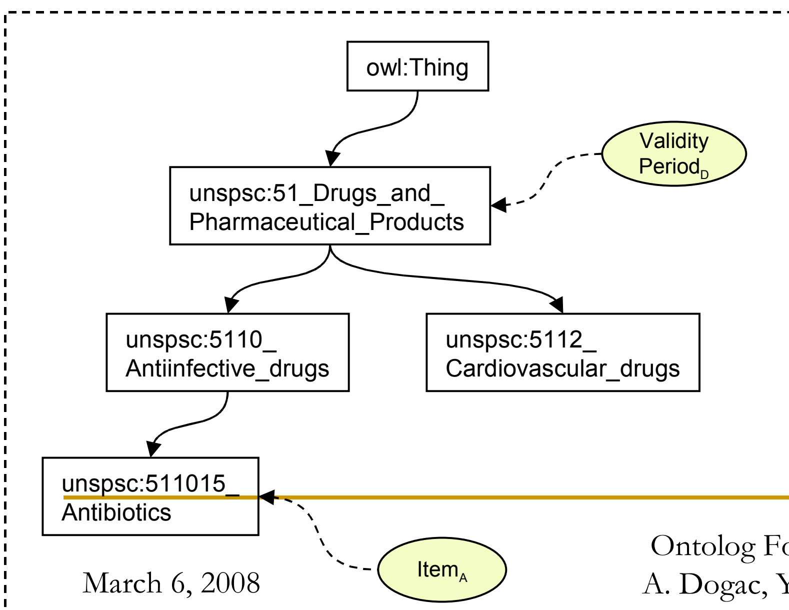
- Similarly, for the context class J, the components "C1", "C2", and "C3" must be merged

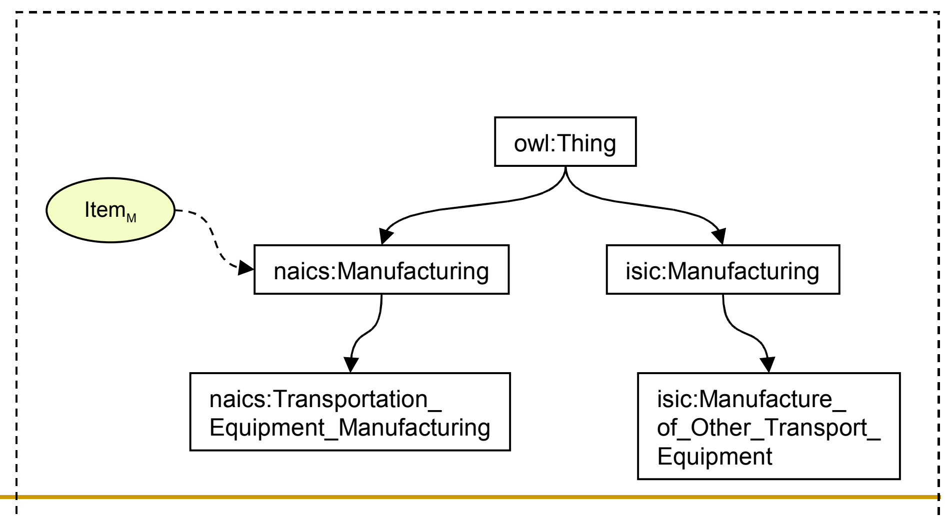


Document Schema Customization Service

- Assume we wish to customize a “catalogue” to “Antibiotic Manufacturing”
- Assume the customized components “ValidityPeriodD”, “itemA” and “itemM” are annotated using respective context ontology classes
- The Customized “catalogue” contains the components “ValidityPeriodD”, and a merged version of “itemA” and “itemM”





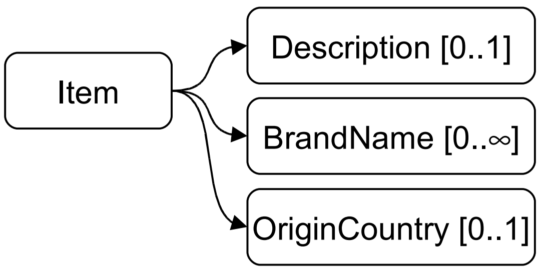


Component Merge Service

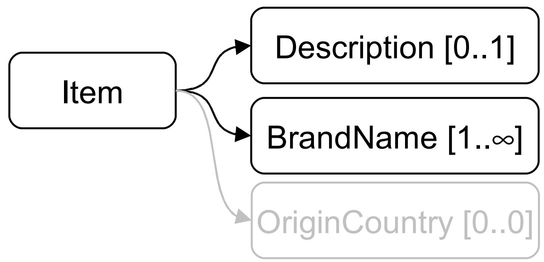
- Given multiple custom versions of a component, generates a combined version
  - Derivation operations (extensions and restrictions) are extracted from individual versions
  - Extracted derivations are successively added to the base version
- Resulting component is a valid specialization of all versions in terms of UBL validation

Component Merge Service

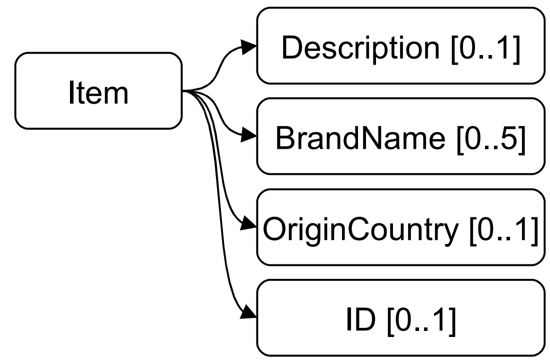
Original Component



Custom Component 1



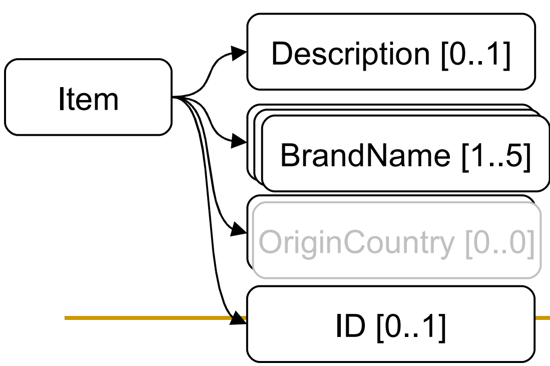
Custom Component 2



BrandName ⇒ [1..∞]
OriginCountry ⇒ [0..0]

BrandName ⇒ [0..5]
ID ⇒ [0..1]

Merged Component



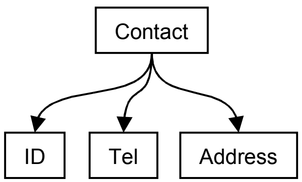
BrandName ⇒ [1..∞]
OriginCountry ⇒ [0..0]
BrandName ⇒ [0..5]
ID ⇒ [0..1]

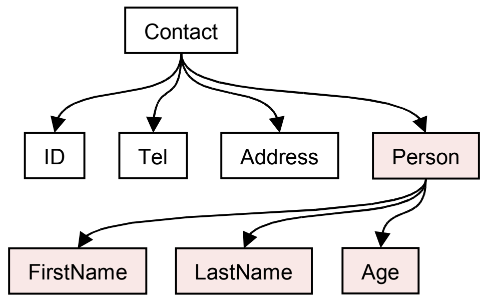
Eliminating Redundancy

- Merging extension operations may cause redundancy in merged component
  - Custom versions may contain the same extension
  - Custom versions may contain structurally different yet semantically similar extensions
- UBL Component Ontology is (to be described later in the talk) utilized to discover semantic redundancy
  - In case of equivalent extensions, only one extension is added to the merged component
  - In case of subsuming extensions, only the extension corresponding to the child class is added

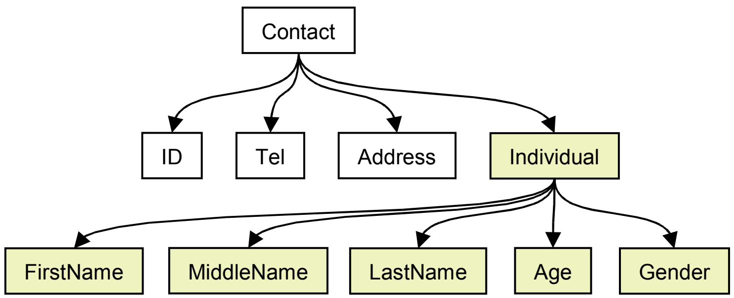
Eliminating Redundancy

- Assume (2) and (3) are merged to yield (4): there is redundancy



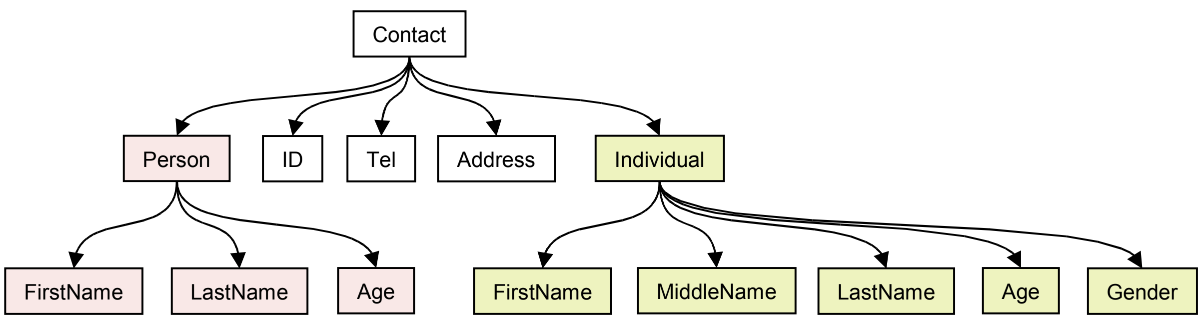


(2)



(3)

- This redundancy is automatically eliminated



(4)

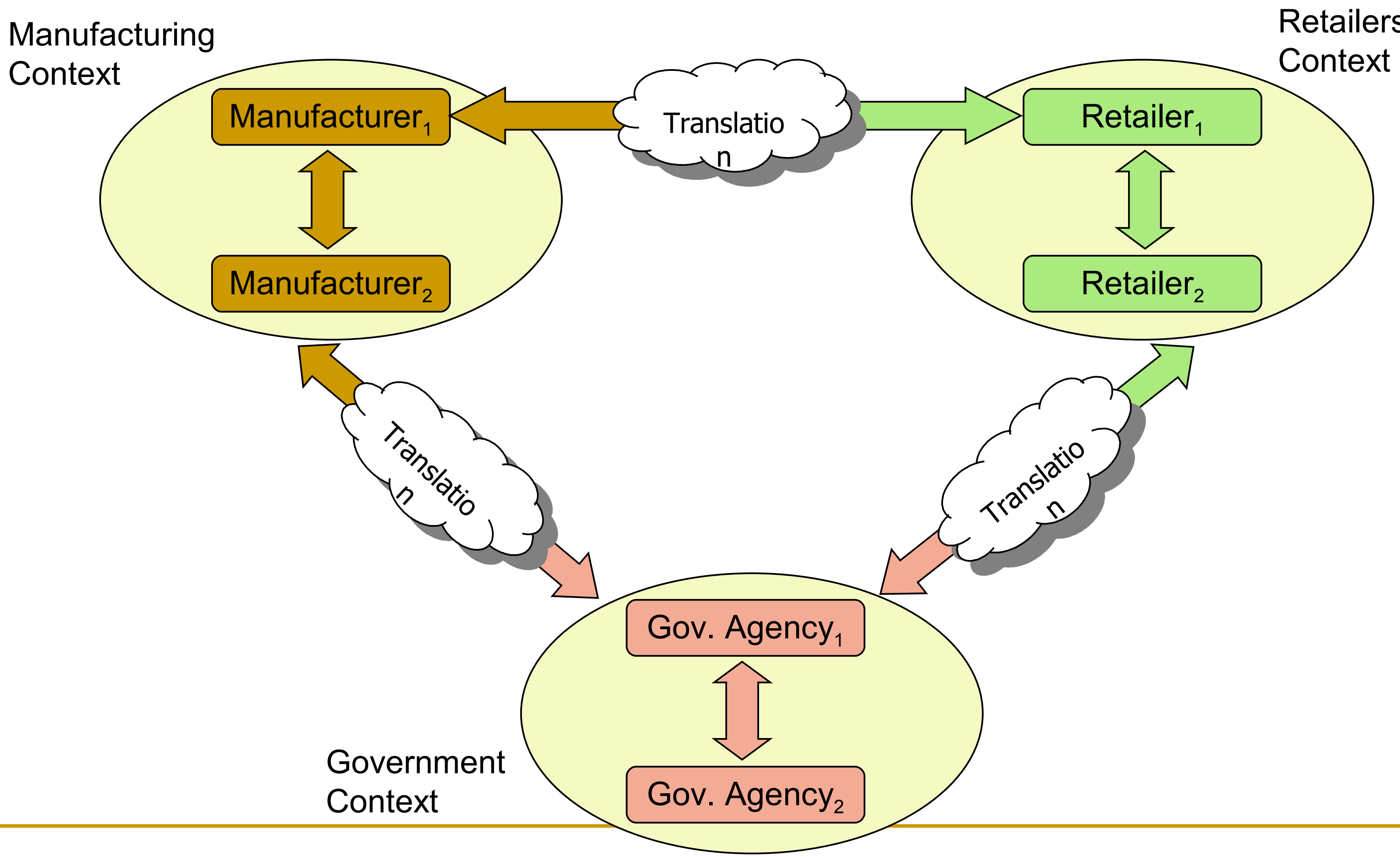
Talk Outline

- A Brief Overview of Electronic Business Document Standards
- UN/CEFACT Core Component Technical Specification
- Semantic Tools for Interoperability Support
  - Use of Ontologies for Semantic Annotation and Ontology Alignment
  - Document Translation
  - System Architecture and Operation
- Conclusions

Motivation: Need for Semantic Interoperability

- Businesses operate in different contexts mandating different rules and regulations for their operations
- Improved customization mechanisms have the potential to encourage more users for tailoring schemas for their needs
- As more users adopt customized schemas, it becomes harder to maintain interoperability among the UBL Community
- A mechanism is required to support interoperability:
  - Individual communities should be free to adopt schemas that best suit their specific needs
  - Members of different communities should not need to know each others' schemas in order to make business

UBL Communities

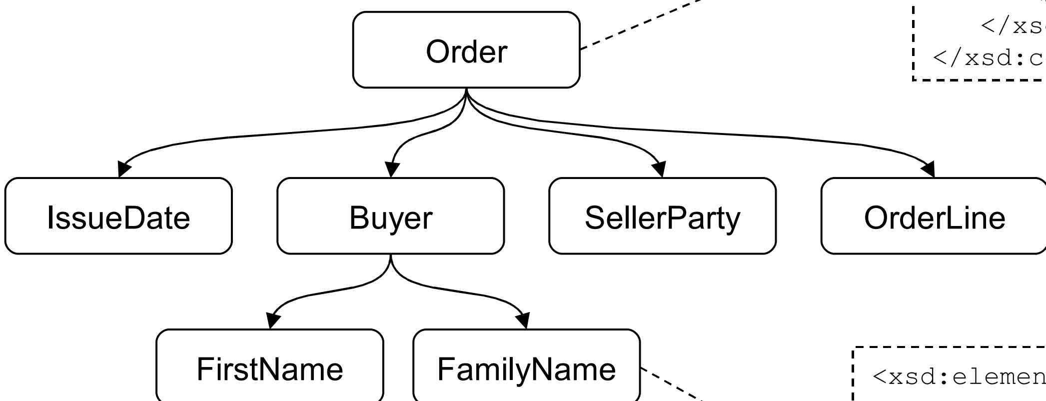


Semantic Translation Mechanism

- A semantic translation mechanism is developed
- This mechanism is based on a UBL Component Ontology which represents structure and semantics of components
- Component Ontology is processed by reasoners to compute further relationships between components
- These relationships are interpreted to adapt document content between different schemas

UBL Components

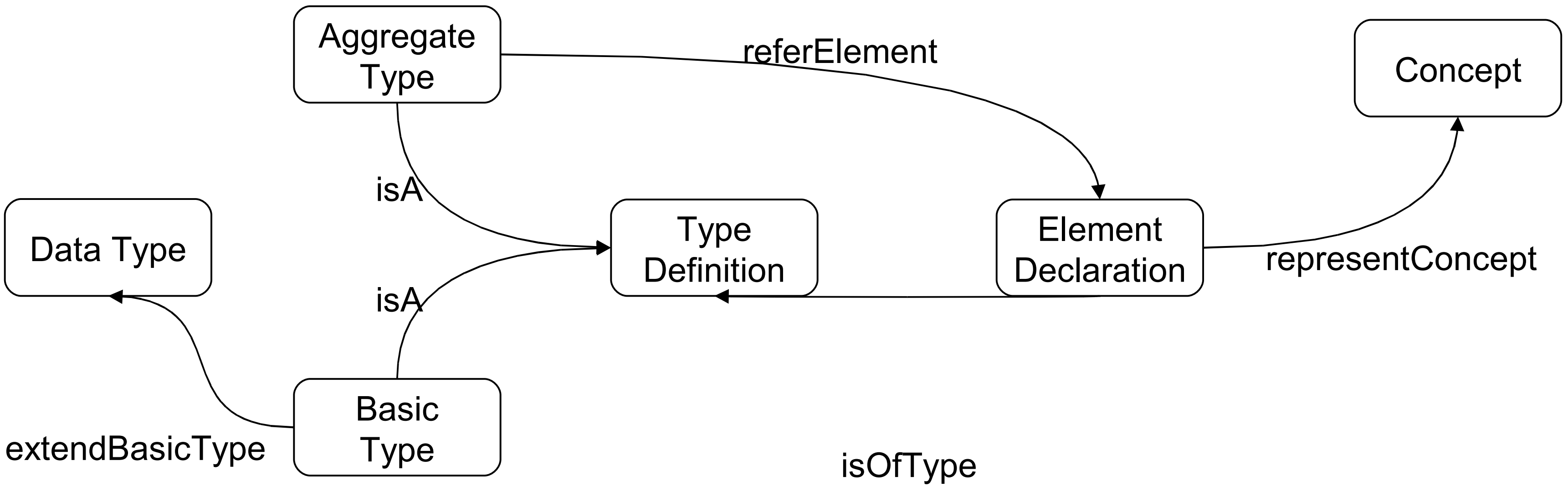
```
<xsd:element name="Order" type="OrderType" />

<xsd:complexType name="OrderType">
   <xsd:sequence>
      <xsd:element ref="IssueDate" />
      <xsd:element ref="Buyer" />
      <xsd:element ref="SellerParty" />
      <xsd:element ref="OrderLine" />
   </xsd:sequence>
</xsd:complexType>
```




```
<xsd:element name="FamilyName" type="FamilyNameType" />

<xsd:complexType name="FamilyNameType">
   <xsd:simpleContent>
      <xsd:extension base="udt:NameType"/>
   </xsd:simpleContent>
</xsd:complexType>
```

UBL Component Ontology

Business concepts such as PostalAddressConcept, DeliveryAddressConcept, specifying concepts represented by UBL components



UBL Component Ontology

- Classes are defined in terms of relations with other classes
- Existential restriction construct of OWL is used to specify those relations
  - aBasicType ≡ (BasicType ∩ (∃extendBasicType. aDataType))
  - anAggregateType ≡ (AggregateType ∩ (∃referElement. ($anElement_1$ ∩ .. ∩ $anElement_n$)))
  - anElement ≡ (ElementDeclaration ∩ ∃representConcept. aConcept ∩ ∃isOfType. aType)

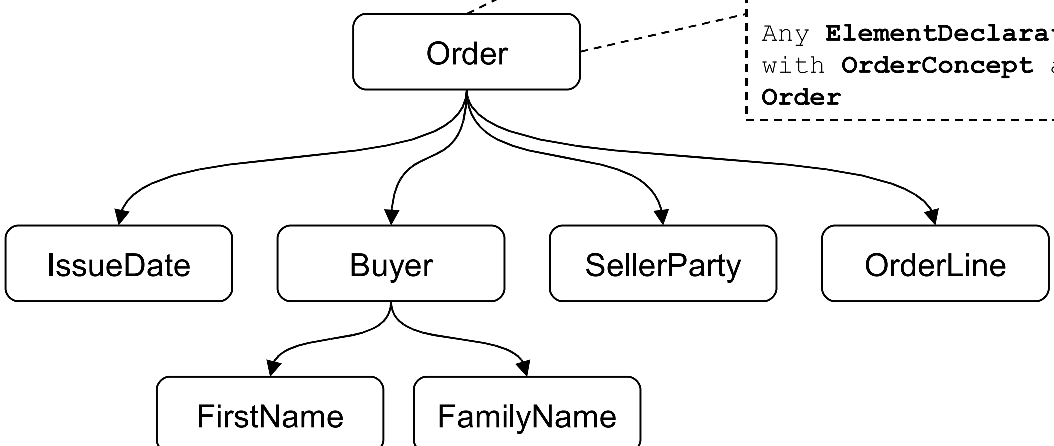
UBL Component Ontology

OrderType ≡ (AggregateType ∩ (∃referElement. (IssueDate ∩ Buyer ∩ *SellerParty* ∩ OrderLine)))

Any **AggregateType** that has **referElement** relationship with **IssueDate** and **Buyer** and **SellerParty** and **OrderLine** is an **OrderType**

Order ≡ (ElementDeclaration ∩ ∃representConcept.OrderConcept ∩ ∃isOfType. OrderType)

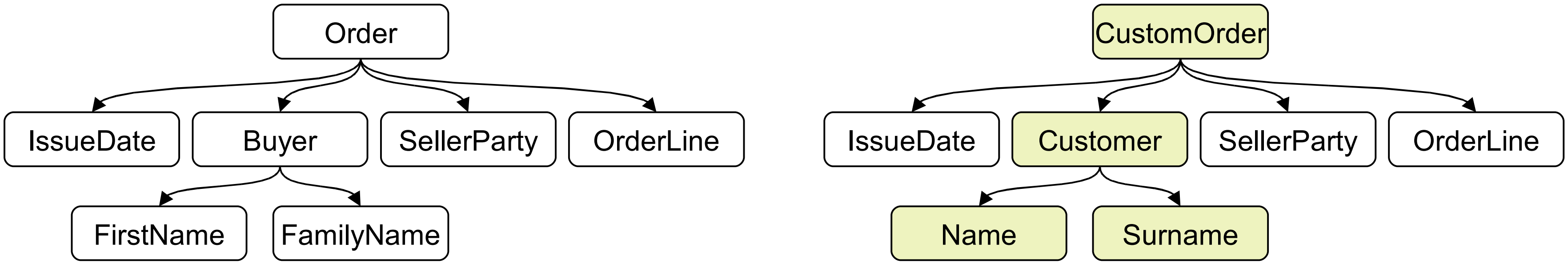
Any **ElementDeclaration** that has a **representConcept** relationship with **OrderConcept** and **isOfType** relationship with **OrderType** is an **Order**



FamilyNameType ≡ (BasicType ∩ (∃extendBasicType. udt:NameType))

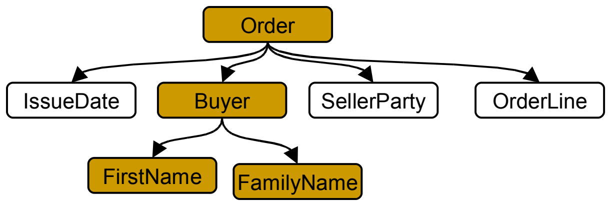
Any **BasicType** that has an **extendBasicType** relationship with **udt:NameType** is a **FamilyNameType**

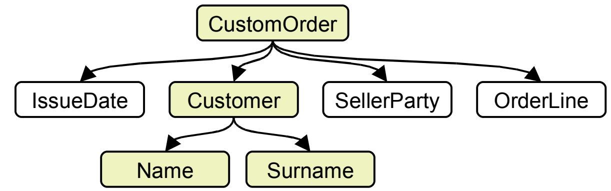
Computing Translations



- For a human being, the similarity between Order and CustomOrder is obvious
- Component Ontology expressions describe components in a machine processable manner so that automated processes can compute the relationship between Order and CustomOrder

Computing Translations





1. Order ≡ (ElementDeclaration ∩ (∃representConcept. OrderConcept) ∩ (∃isOfType. OrderType))
2. OrderType ≡ (AggregateType ∩ (∃referElement. (IssueDate ∩ Buyer ∩ SellerParty ∩ OrderLine)))
3. Buyer ≡ (ElementDeclaration ∩ (∃representConcept. BuyerConcept) ∩ (∃isOfType. PersonType))
4. PersonType ≡ (AggregateType ∩ (∃referElement. (FirstName ∩ FamilyName)))
5. FirstName ≡ (ElementDeclaration ∩ (∃representConcept. FirstNameConcept) ∩ (∃isOfType. FirstNameType))
6. FirstNameType ≡ (BasicType ∩ (∃extend. TextType))
7. FamilyName ≡ (ElementDeclaration ∩ (∃representConcept.FamilyNameConcept) ∩ (∃isOfType.FamilyNameType))
8. FamilyNameType ≡ (BasicType ∩ (∃extend. TextType))

9. CustomOrder ≡ (ElementDeclaration ∩ (∍representConcept. OrderConcept) ∩ (∍isOfType. CustomOrderType))
10. CustomOrderType ≡ (AggregateType ∩ (∍referElement.(IssueDate ∩ Customer ∩ SellerParty ∩ OrderLine)))
11. Customer ≡ (ElementDeclaration ∩ (∍representConcept. BuyerConcept) ∩ (∍isOfType. CustomPersonType))
12. CustomPersonType ≡ (AggregateType ∩ (∍referElement.(Name ∩ Surname)))
13. Name ≡ (ElementDeclaration ∩ (∍representConcept. FirstNameConcept) ∩ (∍isOfType. NameType))
14. NameType ≡ (BasicType ∩ (∍extend. TextType))
15. Surname ≡ (ElementDeclaration ∩ (∍representConcept. FamilyNameConcept) ∩ (∍isOfType. SurnameType))
16. SurnameType ≡ (BasicType ∩ (∍extend. TextType))

17. FirstNameType ≡ NameType *(6 and 14)*
18. FirstName ≡ Name *(5, 13 and 17)*
19. FamilyNameType ≡ SurnameType *(8 and 16)*
20. FamilyName ≡ Surname *(7, 15 and 19)*
21. PersonType ≡ CustomPersonType *(4, 12, 18 and 20)*
22. Buyer ≡ Customer *(3, 11 and 21)*
23. OrderType ≡ CustomOrderType *(2, 10 and 22)*
24. Order ≡ CustomOrder *(1, 9 and 23)*

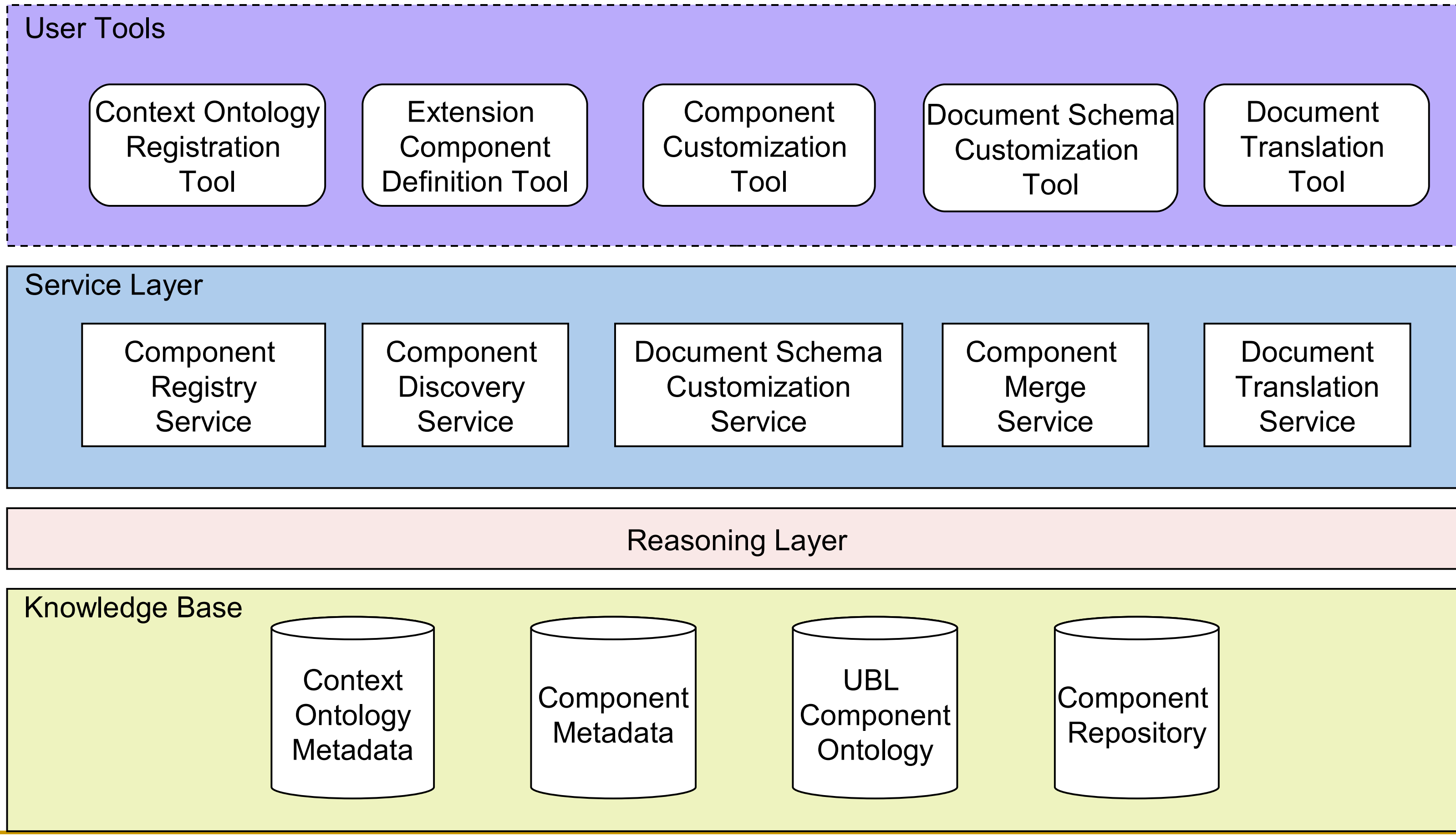
Translatability

- Equivalence relationship between Component Ontology classes is an indication of structural and semantic similarity between corresponding components
  - It is possible to translate content between such components
- Class-subclass relationship between Component Ontology classes is an indication that corresponding components are semantically similar and structurally subsuming
  - It is possible to translate all content from subsuming component to the other, but some of the content cannot be translated back

Talk Outline

- A Brief Overview of Electronic Business Document Standards
- UN/CEFACT Core Component Technical Specification
- Semantic Tools for Interoperability Support
  - Use of Ontologies for Semantic Annotation and Ontology Alignment
  - Document Translation
  - System Architecture and Operation
- Conclusions

System Architecture



Component Registry Service

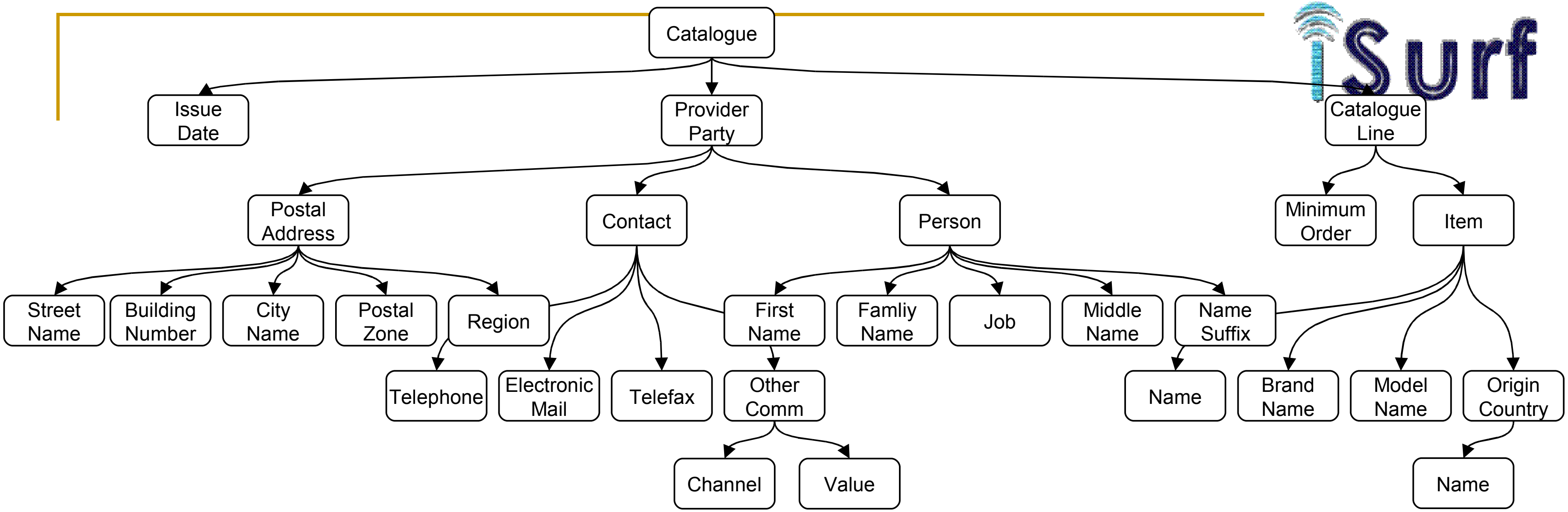
- Component Registry Service maintains knowledge base constructs:
  - Component Repository: XSD definitions for standard, custom and extension components
  - Component Metadata: Metadata definitions in OWL to facilitate component discovery
  - Component Ontology: DL definitions in OWL that support translatability computations

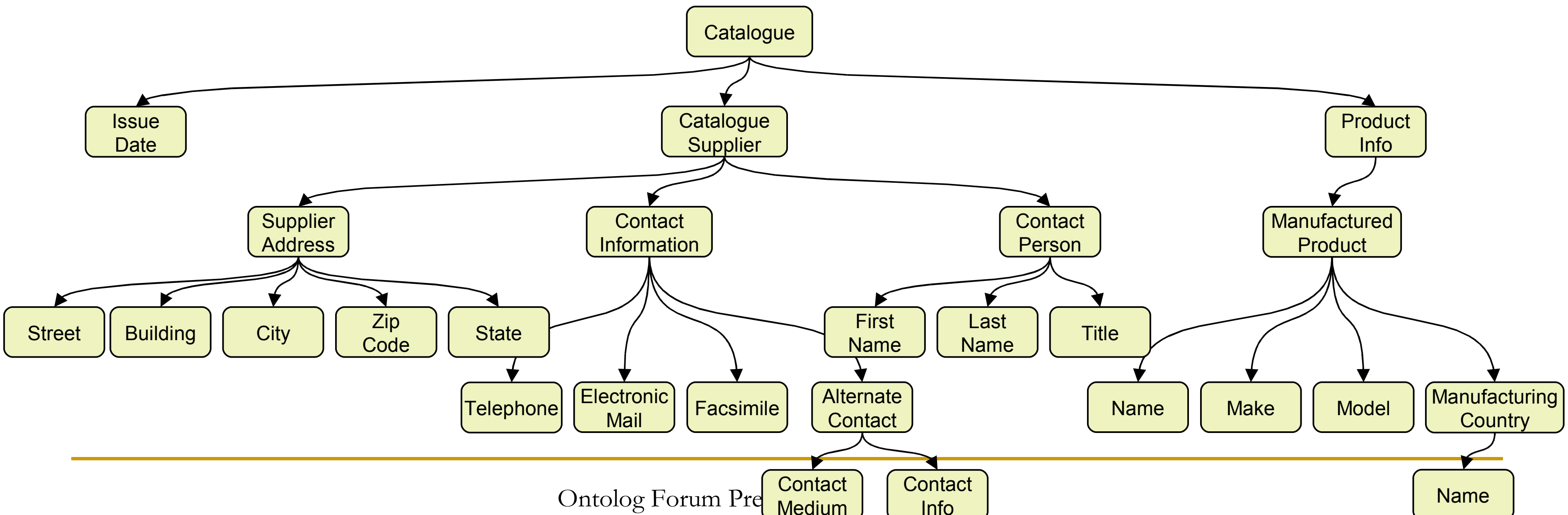
Component Merge Service

- Given multiple custom versions of a component, generates a combined version
  - Derivation operations (extensions and restrictions) are extracted from individual versions
  - Extracted derivations are successively applied to the original component version
- Resulting component is a valid specialization of merged versions in terms of UBL validation

Document Translation Service

- Translation is accomplished by traversing the original document in a top-down manner. For every element:
  - First the corresponding UBL Component is gathered
  - Then the Component Ontology class representing that component is located
  - Then the corresponding Component Ontology class applicable for the target context is computed:
    - First equivalent classes are checked
    - Then sub-classes are checked
    - Finally super-classes are checked
  - If an applicable component can be computed, a corresponding element is added to the target document
  - If an applicable component cannot be computed, original element is added to the UBLExtension hierarchy of the target document


Catalogue
Issue Date
Provider Party
Catalogue Line
Postal Address
Contact
Person
Minimum Order
Item
Street Name
Building Number
City Name
Postal Zone
Region
First Name
Famliy Name
Job
Middle Name
Name Suffix
Telephone
Electronic Mail
Telefax
Other Comm
Name
Brand Name
Model Name
Origin Country
Channel
Value
Name


Catalogue
Issue Date
Catalogue Supplier
Product Info
Supplier Address
Contact Information
Contact Person
Manufactured Product
Street
Building
City
Zip Code
State
First Name
Last Name
Title
Telephone
Electronic Mail
Facsimile
Alternate Contact
Name
Make
Model
Manufacturing Country
Contact Medium
Contact Info
Name

```xml
<Catalogue>
    <IssueDate>2007-12-15+03:00</>
    <ProviderParty>
        <PostalAddress>
            <StreetName>62nd Avenue South</>
            <BuildingNumber>CC-206</>
            <CityName>Kent</>
            <PostalZone>98032</>
            <Region>WA</>
        </PostalAddress>
        <Contact>
            <Telephone>+1 253 854 3237</>
            <ElectronicMail>TireCollection@GoodTires.com</>
            <Telefax>+1 253 854 3239</>
            <OtherCommunication>
                <Channel>Mobile Phone</>
                <Value>+1 253 324 5654</>
            </OtherCommunication>
        </Contact>
        <Person>
            <FirstName>Ben</>
            <FamilyName>Clark</>
            <Job>Sales Officer</>
            <MiddleName>Johnson</>
            <NameSuffix>Mr.</>
        </Person>
    </ProviderParty>
    <CatalogueLine>
        <Item>
            <Name>Winter Tire</>
            <BrandName>PR-854</>
            <ModelName>Pirelli</>
            <OriginCountry>
                <Name>Turkey</>
            </OriginCountry>
        </Item>
    </CatalogueLine>
</Catalogue>
```

```xml
<Catalogue>
    <UBLExtension>
        <ProviderParty>
            <Person>
                <MiddleName>Johnson</>
                <NameSuffix>Mr.</>
            </Person>
        </ProviderParty>
    </UBLExtension>
    <IssueDate>2007-12-15+03:00</>
    <CatalogueSupplier>
        <SupplierAddress>
            <Street>62nd Avenue South</>
            <Building>CC-206</>
            <City>Kent</>
            <ZipCode>98032</>
            <State>WA</>
        </SupplierAddress>
        <ContactInformation>
            <Telephone>+1 253 854 3237</>
            <ElectronicMail>TireCollection@GoodTires.com</>
            <Facsimile>+1 253 854 3239</>
            <AlternateContactInfo>
                <ContactMedium>Mobile Phone</>
                <ContactInfo>+1 253 324 5654</>
            </AlternateContactInfo>
        </ContactInformation>
        <ContactPerson>
            <FirstName>Ben</>
            <LastName>Clark</>
            <Title>Sales Officer</>
        </ContactPerson>
    </CatalogueSupplier>
    <ProductInfo>
        <ManufacturedProduct>
            <Name>Winter Tire</>
            <Make>PR-854</>
            <Model>Pirelli</>
            <ManufacturingCountry>
                <Name>Turkey</>
            </ManufacturingCountry >
        </ManufacturedProduct>
    </ProductInfo>
</Catalogue>
```

Talk Outline

- A Brief Overview of Electronic Business Document Standards
- UN/CEFACT Core Component Technical Specification
- Semantic Tools for Interoperability Support
  - Use of Ontologies for Semantic Annotation and Ontology Alignment
  - Document Translation
  - System Architecture and Operation
- Conclusions

Conclusion

- Specific contributions of our work:
  - Annotation of components using classes from context ontologies
  - Development of context ontologies for the formal representation of business context domains
  - Facilitating the discovery, reuse and customization of components
  - Development of a component ontology to represent structure and semantics of components
  - Utilization of the ontology for the computation of similarities between components
  - Providing a prototype implementation for the realization of our approach

Thank you very much for your attention!
Questions?

Extra Slides: Improving the Performance of the Translation Process

UBL Component Ontology

- UBL aggregate types are composed of numerous elements
- Not all elements are significant for determining translatability
  - All mandatory elements are considered significant and automatically defined in component ontology expressions
  - It is expected from users to specify which optional elements are to be considered as significant for translatability computations

UBL Component Ontology

```
<xsd:complexType name="EndorsementType">
   <xsd:sequence>
     <xsd:element ref="DocumentID" minOccurs="1" maxOccurs="1"/>
     <xsd:element ref="ApprovalStatus" minOccurs="1" maxOccurs="1"/>
     <xsd:element ref="Remarks" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="EndorserParty" minOccurs="1" maxOccurs="1"/>
     <xsd:element ref="Signature" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
 </xsd:complexType>
```

**EndorsementType ≡ (AggregateType ∩
∃referElement.(DocumentID ∩ ApprovalStatus ∩ EndorserParty))**

- This allows translatability computations to consider only significant elements
  - Improves outcome and performance of translatability computations