

# Ontologizing EDIFACT Semantics Phase 2



doug foxvog -- NUIG  
TripCom Consortium Meeting  
October 1<sup>st</sup> 2007



- Ontologizing Code Sets
- Ontologizing Data Element/Segment Meaning
- Ontologizing Statement Templates
- Ontology (File) Hierarchy

- Current Task
  - Creating ontologies for terms in Code Sets
- Files
  - One external term ontology file per Data Element or cluster of Data Elements
  - One code set mapping file per Data Element or cluster of Data Elements
  - EDIFACTcodeSets.wsml for defining Code Sets
  - Previously created EDI\_Semantics.wsml
- References
  - Text files created (DE\_nnnn.txt) while ontologizing syntax
  - Source web pages for these data elements
  - OpenCyc for existing terminology
    - <http://www.OpenCyc.org> (download)
    - <http://www.CycFoundation.org/concepts>

- Define concepts, relations, or relations for codes
  - Attributes cannot be referenced.
- Try to have all terms for a Data Element as same type of thing
- Reuse terms in ontology if possible
- Use existing terms if possible
  - ... from other Data Element ontologies
  - ... from OpenCyc
- Create terms for natural super concepts/relations
  - Link components to these terms
- Annotate meaning of each term
  - with `dc:description`

- Create instance of `edis#CodeSet`
  - in `EDIFactCodeSets.wsm1`
  - Different versions for subset-specific codes
- Create ontology file for each code set or related group of code sets
- Create `...Codes` knowledge base file for each ontology file
  - Import ontology into `...Codes` file
  - Assign mappings in `...Codes` file

```
relationInstance edis#encodedAs(mf#NotAmendedByBuyer ,  
    ecs#DE_1129CodeSet, "11")
```

```
relationInstance edis#encodedAs(mf#NoAdvice,  
    ecs#DE_1129SCodeSet, "12E")
```

# Example Referenced Terms Ontology



```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"

namespace {_"http://deri.org/ont/MessageFunction#",
          ack _"http://deri.org/ont/Acknowledgement#",
          cyc _"http://sw.cyc.com/2006/07/27/cyc/#",
          dc _"http://purl.org/dc/elements/1.1#" }

ontology _"http://deri.org/ont/MessageFunction"
. . .
concept cyc#Individual
concept cyc#TemporalThing subConceptOf cyc#Individual
concept cyc#Event subConceptOf cyc#TemporalThing
concept cyc#Action subConceptOf cyc#Event
. . .
concept cyc#FindingSomething subConceptOf cyc#Action
. . .

concept LineItemFoundAbsent subConceptOf {cyc#FindingSomething, DocumentAction}
  nonFunctionalProperties
    dc#description hasValue "A line item is found not to be in the referenced message."
  endNonFunctionalProperties
//      "10"                Not found
// see cyc#Absent

instance ReportingItemDetailsIncluded subConceptOf DocumentAction
  nonFunctionalProperties
    dc#description hasValue "Reporting item details are included."
  endNonFunctionalProperties
//      "11E"                Reporting item details included (SWIFT Code)
```

# Example Code Set Mapping KB



```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"
```

```
namespace {_"http://deri.org/ont/MessageFunctionCodes#",  
          mf _"http://deri.org/ont/MessageFunction#",  
          ecs _"http://deri.org/ont/EDIFACTcodeSets#",  
          edis _"http://deri.org/ont/EDI_Semantics#",  
          cyc _"http://sw.cyc.com/2006/07/27/cyc/#",  
          dc _"http://purl.org/dc/elements/1.1#" }
```

```
ontology _"http:// deri.org/ont/MessageFunctionCodes"
```

```
. . .
```

```
importsOntology _"http://deri.org/ont/MessageFunction"
```

```
relationInstance edis#encodedAs( cyc#WaterSurfaceTransportion ,  
                                ecs#CEFACTRecommendation19CodeSet, "1")
```

```
relationInstance edis#encodedAs( trans#NonContainerVessel ,  
                                ecs#DE_8067CodeSet, "10")
```

```
relationInstance edis#encodedAs( trans#ContainerVessel ,  
                                ecs#DE_8067CodeSet, "11")
```

```
. . .
```

- Ontologizing Code Sets
- Ontologizing Data Element/Segment Meaning
- Ontologizing Statement Templates
- Ontology (File) Hierarchy

- The “meaning” of an instantiation of a Data Element or Data Segment is taken to be an instance, a concept, or a relation.
- Attributes and relation instances restrict the range of meanings for instantiations of message components.
  - Attribute used for relation between two instances.
  - Relation used if ternary
  - Relation used if argument may be concept or relation
- Meanings of instantiations of subcomponents relative to main component can be further restricted.
  - These are meanings of instantiations, not of the format component.

- **componentOfType** – signifies that an instantiation “means” an instance of the specified class.
- **componentOfSubType** – signifies that an instantiation “means” a subclass of the specified class.
- **subcomponentOfType** – signifies that the filler for the Nth position of an instantiation “means” an instance of the specified class.
- **subcomponentOfSubType** – signifies indicating that the filler for the Nth position of an instantiation “means” a subclass of the specified class.

- The meaning of a component is a specific city:

```
relationInstance componentOfSubType (de#DE_3164, cyc#City)
```

- The meaning is a type of medical test:

```
relationInstance componentOfSubType  
(de#DE_4416, cyc#MedicalTesting)
```

- The meaning of the **DTM** segment as the third segment of **APERAK** is a date (not a duration):

```
relationInstance subcomponentOfType  
(mt#APERAKMessage, 3, cyc#Date)
```

- The meaning of a Free Text (**FTX**) segment in this position is a type of dangerous thing:

```
relationInstance subcomponentOfSubType  
(pricat#PRICAT_Loop55, 3, cyc#DangerousTangibleThing)
```

- Ontologizing Code Sets
- Ontologizing Data Element/Segment Meaning
- Ontologizing Statement Templates
- Ontology (File) Hierarchy

- A template is a pattern for generating a single statement from an instantiated data structure.
- A given structure may have multiple templates.
- Classes for templates are:
  - **CodeSet**
  - **ComponentTemplate**
  - **FormulaArgPosition** – position in structure
- Template relations and attributes exist for:
  - Assigning code sets
  - Equating meanings of different message components
  - Specifying functional relations for meanings
  - Collecting a predicate and its arguments for assembling a statement.

- **hasSameMeaningAs** – ternary relation indicating that the meaning for two different argument positions of a structure should be equated.
- **functionalPredicateEncodes** – ternary relation indicating that the meaning for a specified structure is the unique thing related to its filler by the specified predicate, e.g. `hasMedicalID`.
- **directlyEncoded** – attribute that the meaning for a specified structure is the same as what fills it.
- **[ subcomponent ] usesCodeSet** – relation/attribute indicating that the specified code set maps the filler of the specified structure to its “meaning”.

- **templateForComponent** – attribute specifying one of possibly several **ComponentTemplates** for a logical sentence encoded by instantiations of a given information structure.
- **templateRelation** – binary relation specifying the relation for a given template.
- **templateRelationEncoded** – attribute specifying that the relation for a template is encoded in a given structure.
- ...

# Template Attributes / Relations (2)



- **templateArg**<sub><N></sub>**Encoded** (for N 1, 2, or 3) – attribute specifying that the Nth argument for a template is the meaning encoded in the specified FormulaArgPosition of the data structure.
- **templateArg**<sub><N></sub>**Value** – binary relation specifying that the Nth argument for a template is the specified value. The value can be a data value, instance, concept, or relation.
- . . .

- Data Element 8179 uses a two specific code sets:

```
instance mt#DE_8179
  usesCodeSet hasValue ecs#CEFACTRecommendation19CodeSet
  usesCodeSet hasValue ecs#CEFACTRecommendation28CodeSet
  componentOfType hasValue cyc#Conveyance
```

- The third component of the **NAD** data segment is the name of whatever the second component means:

```
instance NADTemplate4 memberOf edis#ComponentTemplate
  templateForComponent hasValue ds#NAD_DS
  templateRelation hasValue cyc#nameStrings
  templateArg1Encoded hasValue edis#SecondSubcomponentPosition
  templateArg2Encoded hasValue edis#ThirdSubcomponentPosition

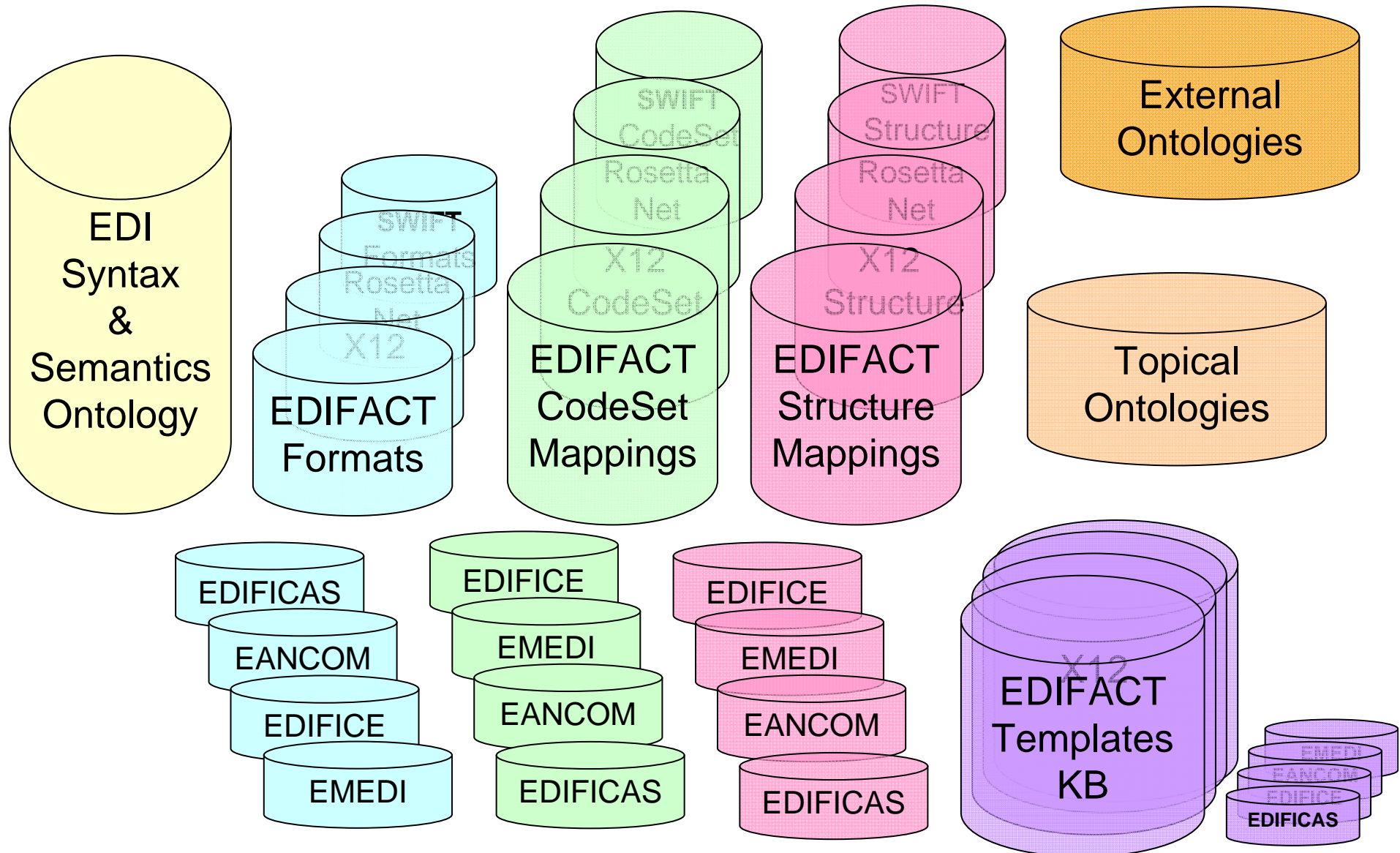
relationInstance templateRelation
  (NADTemplate4 , cyc#nameString)
```

- **templateSubcomponentMustBeOfType** – ternary relation requiring that for a template to be used, the Nth slot of the component it applies to must be an instance of the specified type.
- **templateSubcomponentMustBeOfSubType** – ternary relation requiring that for a template to be used, the Nth slot of its component must be a subconcept of the specified type.
- **templateSubcomponentMustBeRelation** – attribute requiring that for a template to be used, the Nth slot of its component must be a relation.

- Ontologizing Code Sets
- Ontologizing Data Element/Segment Meaning
- Ontologizing Statement Templates
- **Ontology (File) Hierarchy**

- Ontologies and their files are of several types **Pre-TripCom**
- Defining terminology for encoding EDI messages
  - Defining EDIFACT message syntax **M6-12**
  - Defining concepts, instances, and relations which EDIFACT messages discuss **M13-18 ff.**
  - Mapping of codes in EDIFACT code sets to their meanings **M13-18 ff.**
  - Mapping EDIFACT structures to the type of thing their instantiations mean **M19-24**
  - Defining templates for sentences implicit in EDIFACT messages **M19-24**

# Ontology / KB (File) Types



## EDI terminology ontologies/files

- Do not need to access any other ontologies

## EDIFACT message syntax ontologies/files

- Need access only to EDI terminology ontologies
- Have two levels
  - For general EDIFACT syntax
  - For subset-specific syntax
- Subset-specific files inherit from general EDIFACT syntax files, not from other subsets
- Some subset-specific files need sibling access

- Do not need EDI-specific ontologies
- Reference external ontology terms
- Can be consolidated if concerning related topics
- Could inherit core “upper ontology”
- Imported by
  - Knowledge bases mapping code set codes
  - Knowledge bases mapping EDIFACT structure meanings
  - Knowledge bases defining templates
- Not created in EDIFACT subset-specific forms

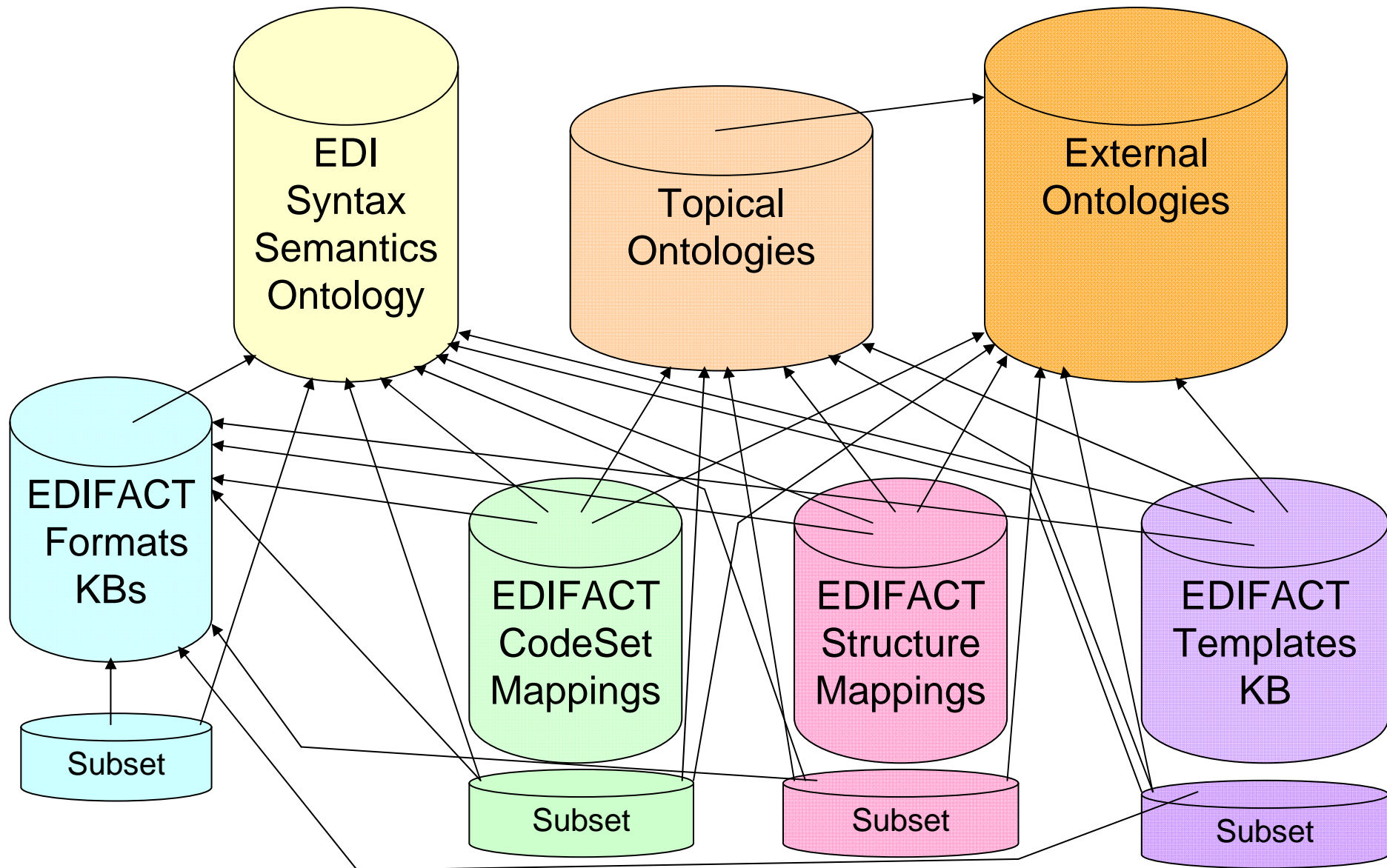
M19-24

- Created for individual Data Elements or closely-related set of Data Elements.
- Do not need EDI-format knowledge bases
- Import referenced things ontologies
- Do not need to access template ontologies
- Not imported by other knowledge bases
- Optionally:
  - created in EDIFACT subset-specific forms
  - include mappings for several EDIFACT subsets

- Collect basic meaning of multiple Data Segments or Data Elements
- Created at two levels
  - for EDIFACT generally
  - for individual EDIFACT subsets
- Reference EDIFACT syntax knowledge bases
- Import referenced things ontologies
- Do not need Code Set knowledge bases
  - Except to reference when creating
- Do not need to access template ontologies
- Not imported by other knowledge bases

- Created for Data Segments and EDI Message Types; occasionally for Complex Data Segments
- Import referenced things ontologies
- Reference EDIFACT syntax knowledge bases
- Do not need to access structure meaning KBs
- Not imported by other knowledge bases
- Created at different levels:
  - EDIFACT subset-specific level
  - Generic EDIFACT level

# Ontology / KB (File) Types



# Questions?