

Part 3 – UML Profile for Core Components (UPCC 3.0)

Modeling Core Components using UML

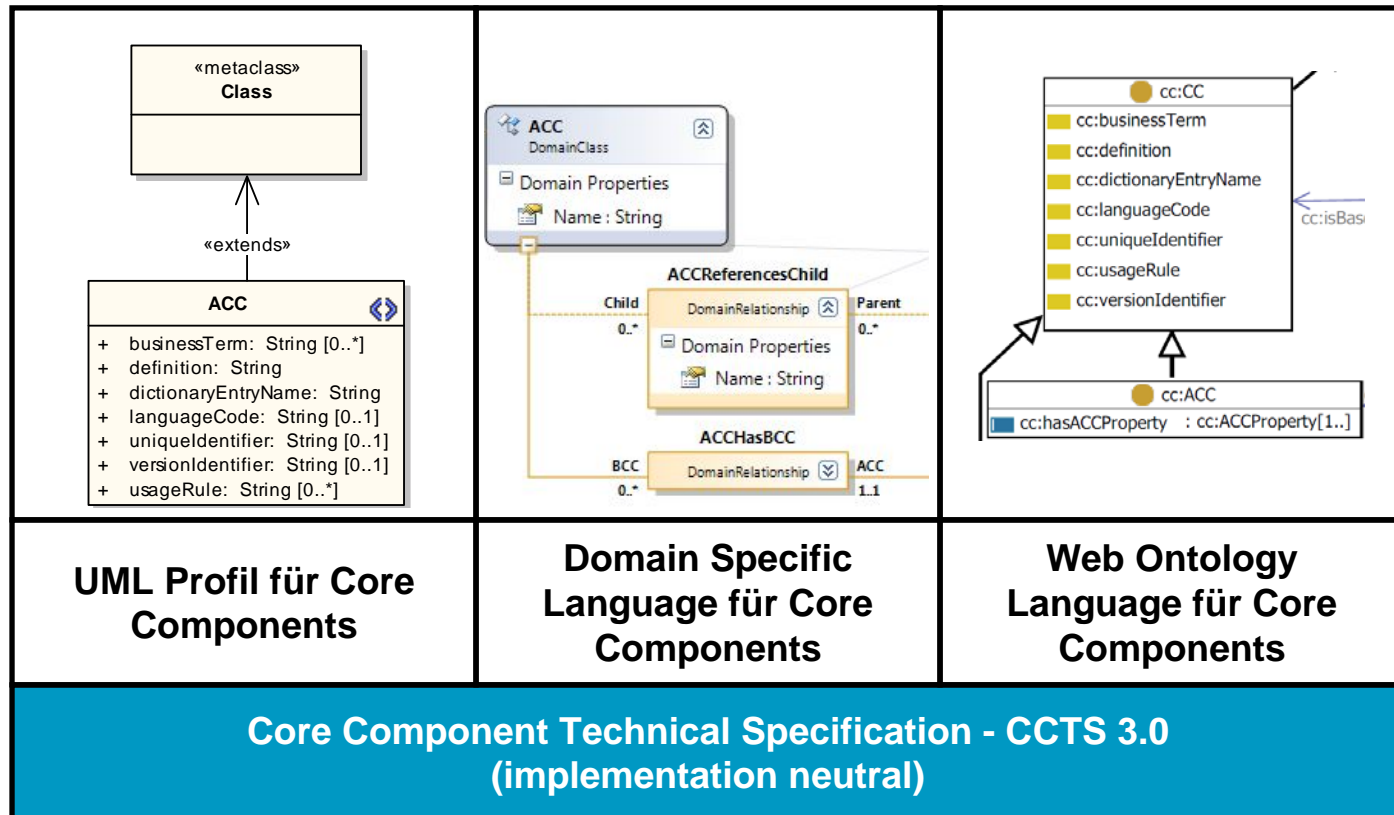
Research Studio Inter-Organisational Systems
Project Public Private Interoperability

Agenda

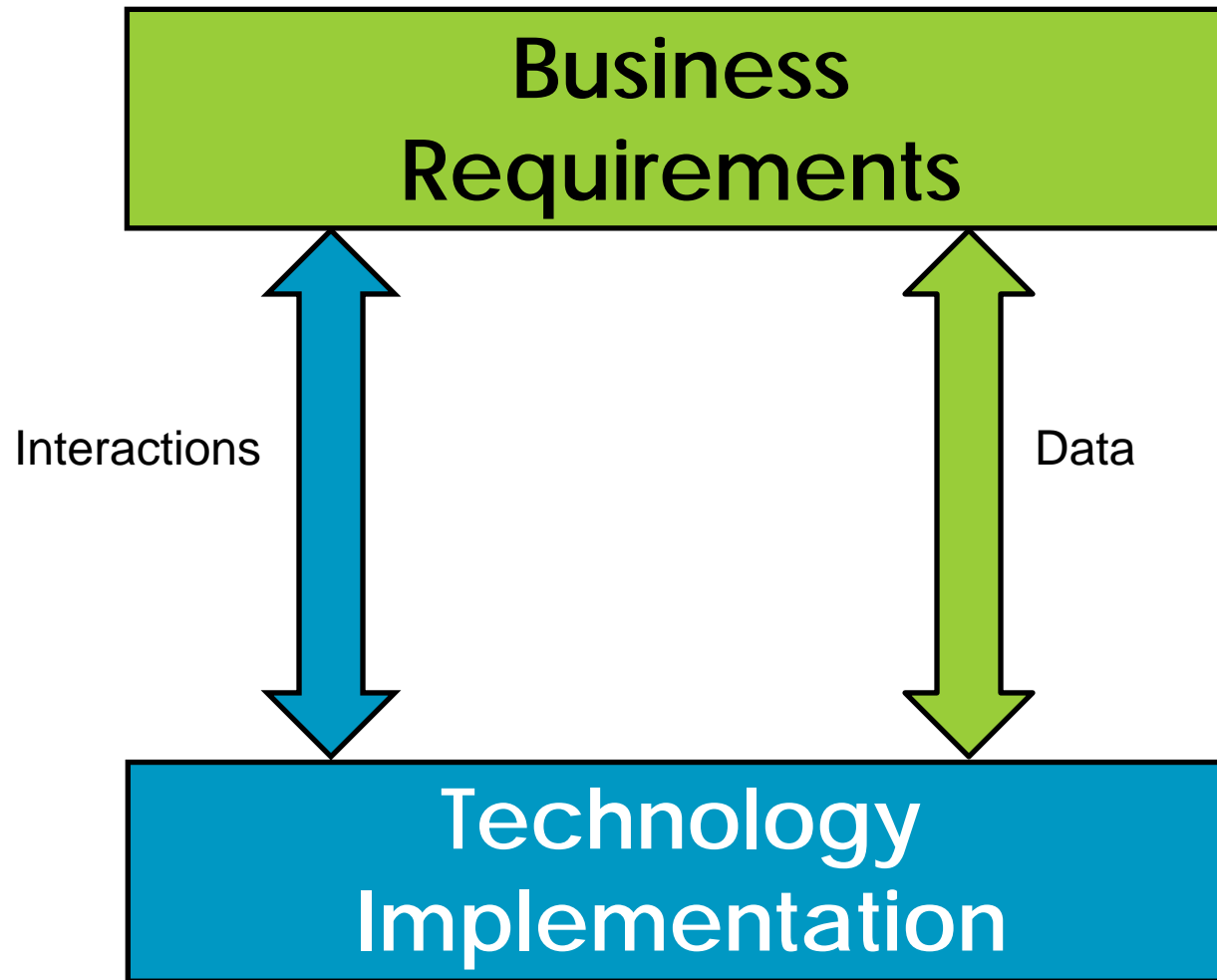
- Introduction
- The idea behind a UML Profile
- The different views of the UPCC
- From UML to XML
- Configuring IT systems with generated XML information

Shortcomings of core components

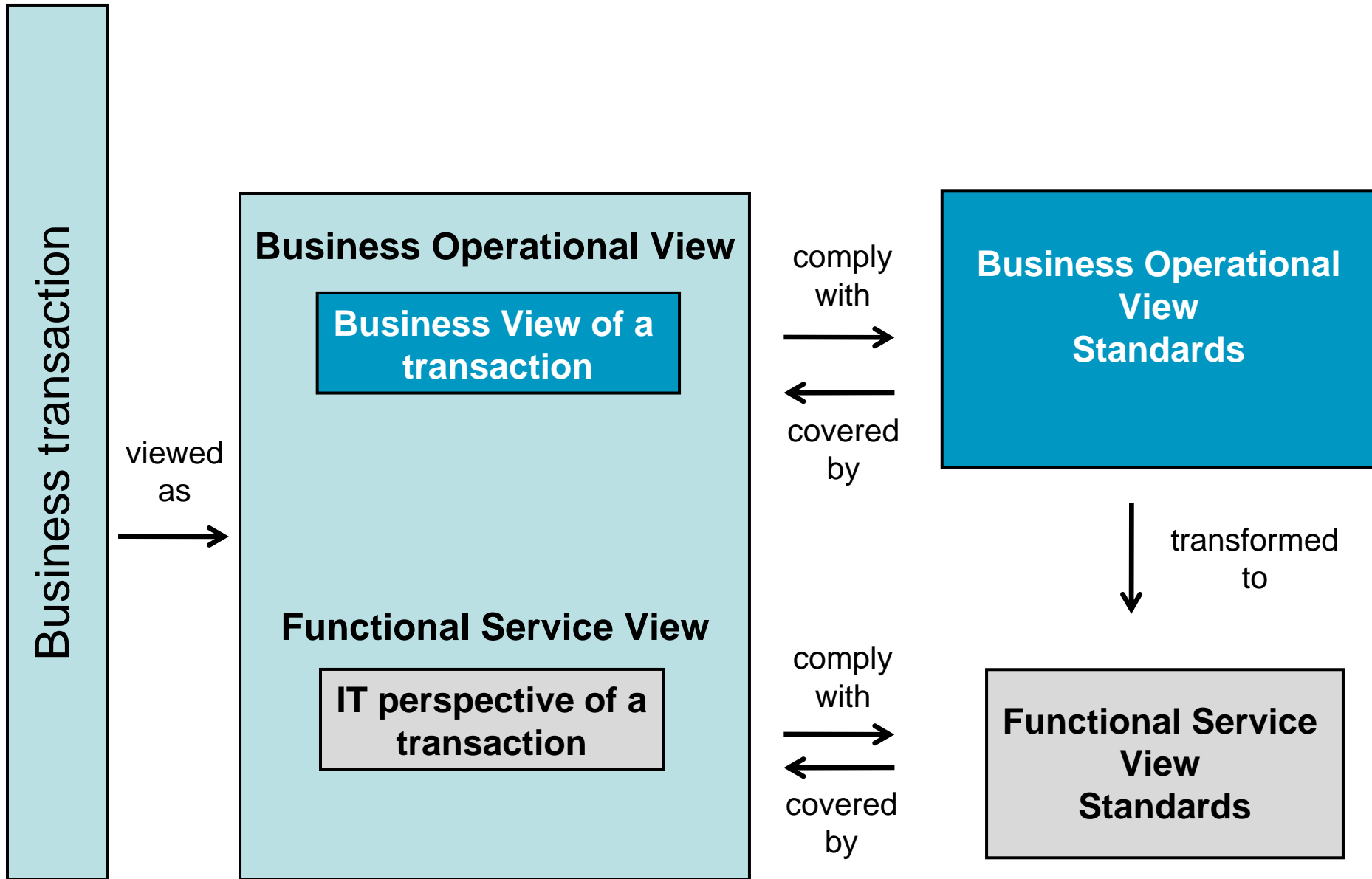
- No formal representation mechanism available for core components
- No direct integration in modeling tools available
- Core components are defined in an implementation neutral manner



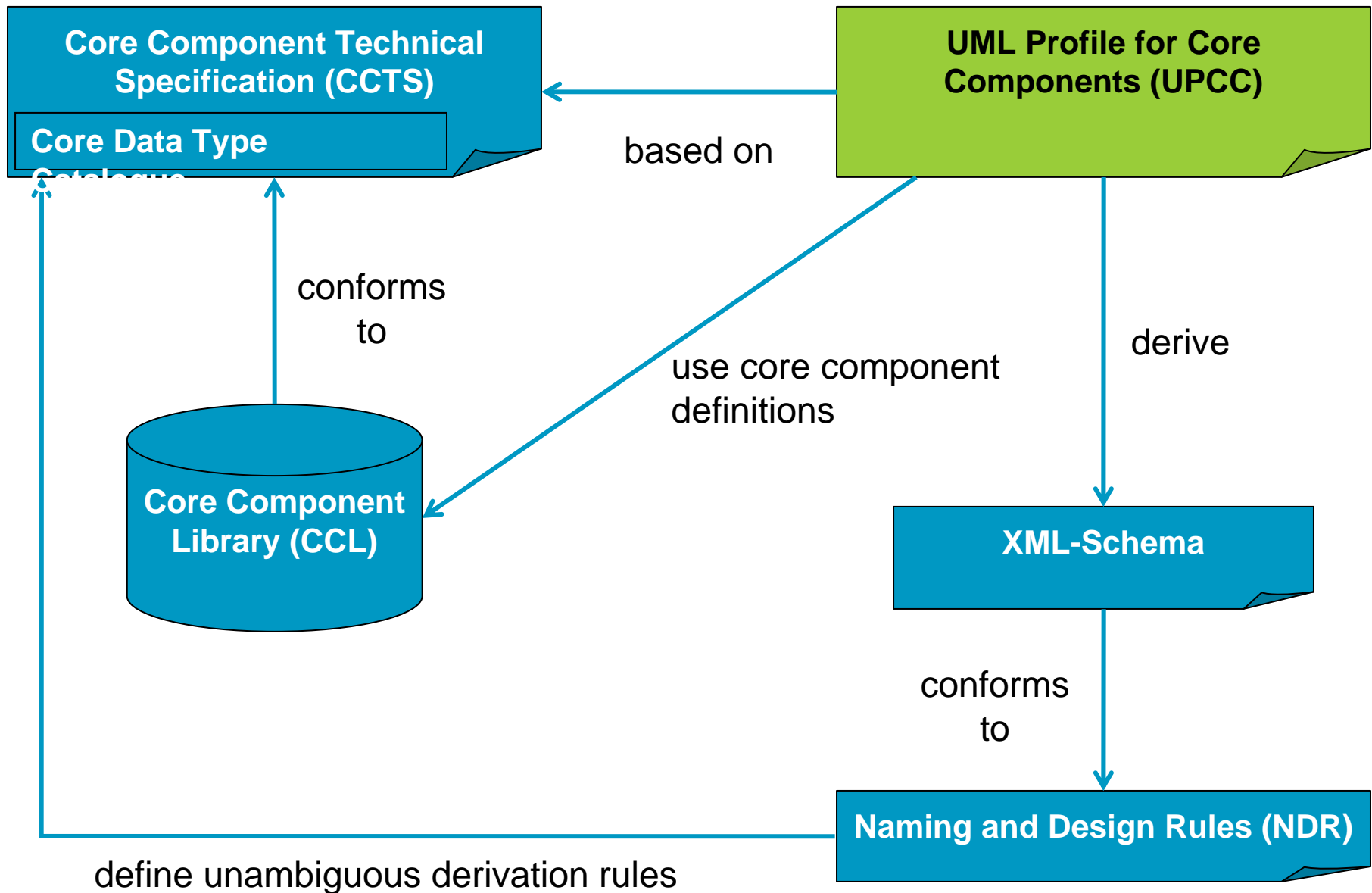
Requirements vs. Technology



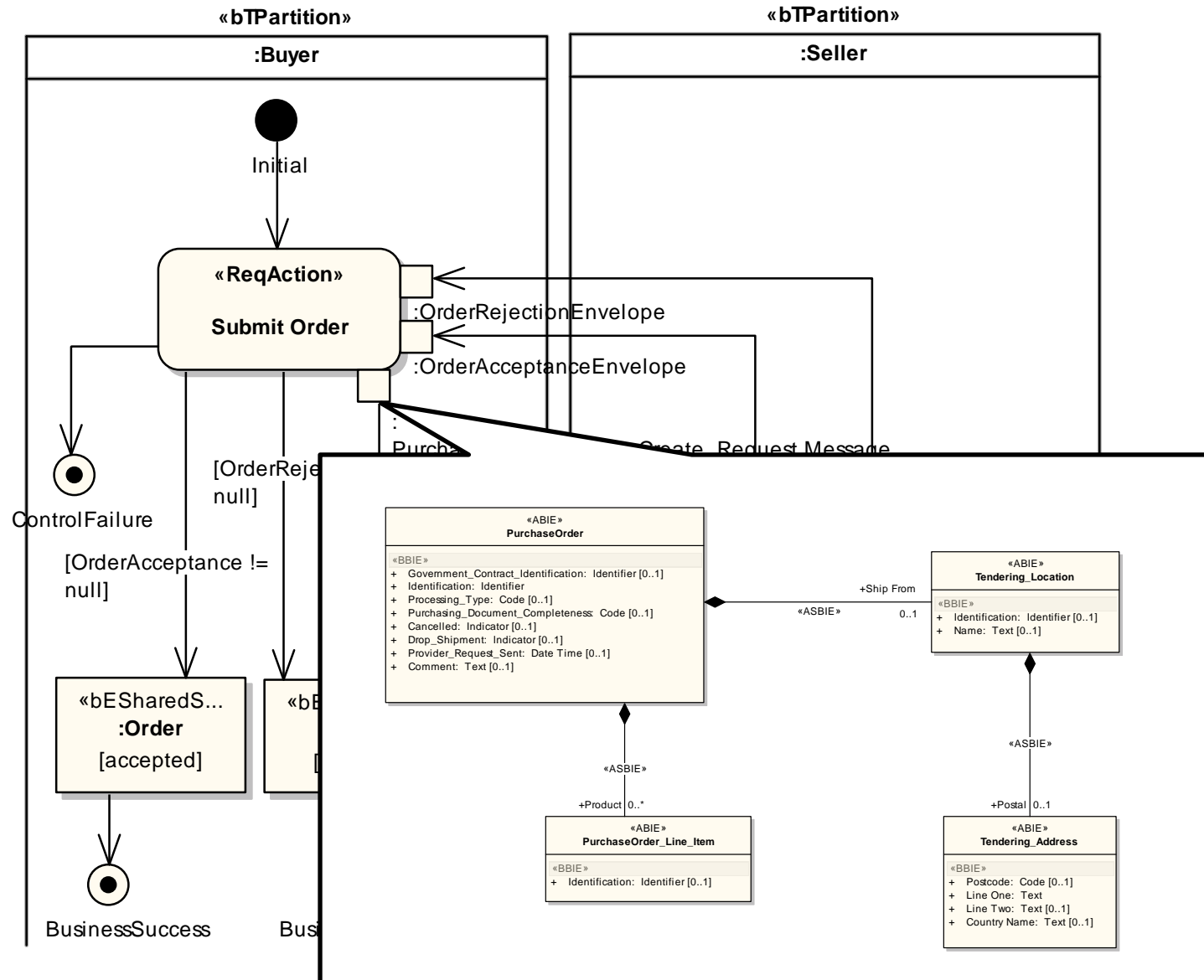
Open-edi Reference Model – IO 14662



Overview of Core Component related UN/CEFACT specifications



Business documents in UN/CEFACT's Modeling Methodology

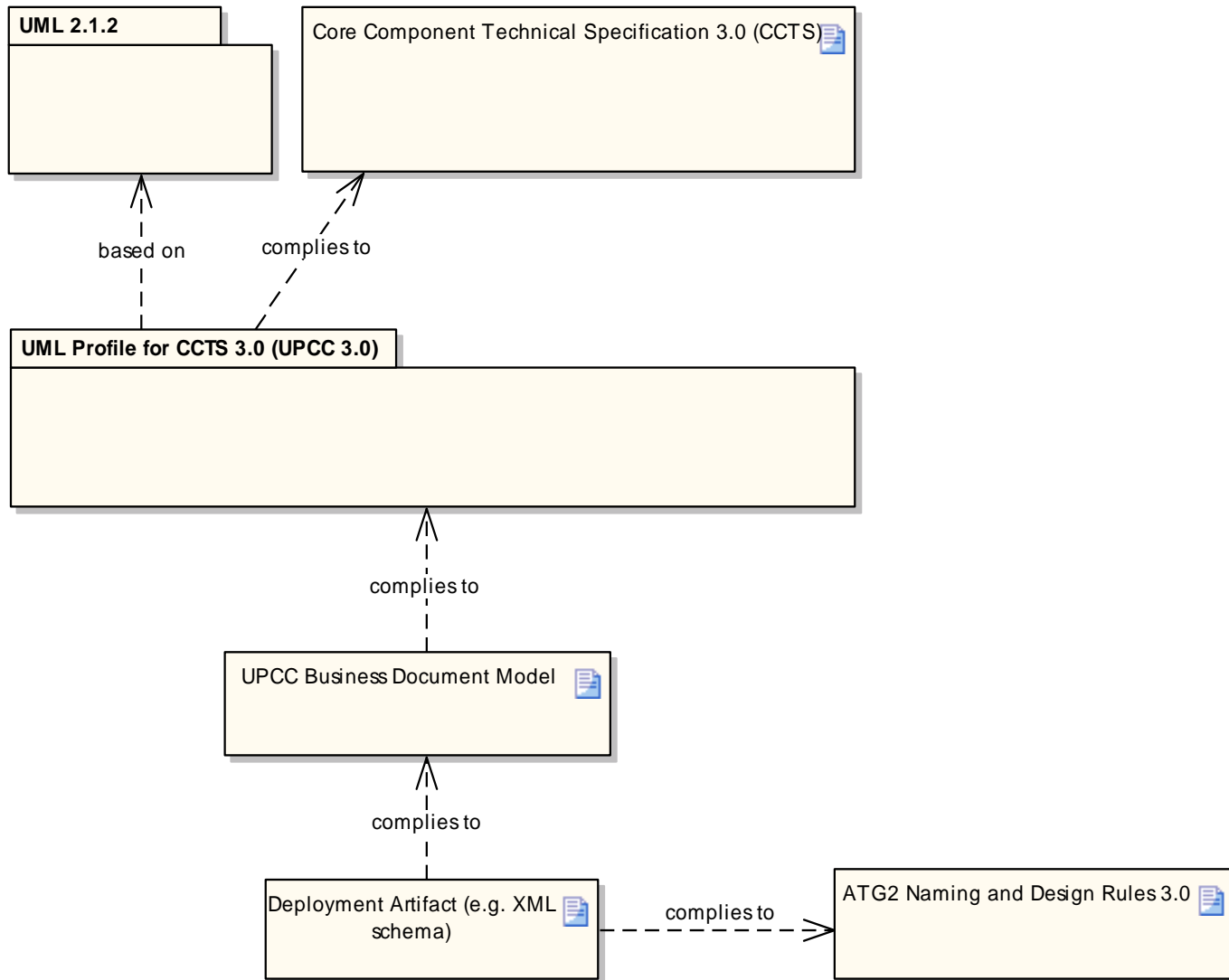


Health warning

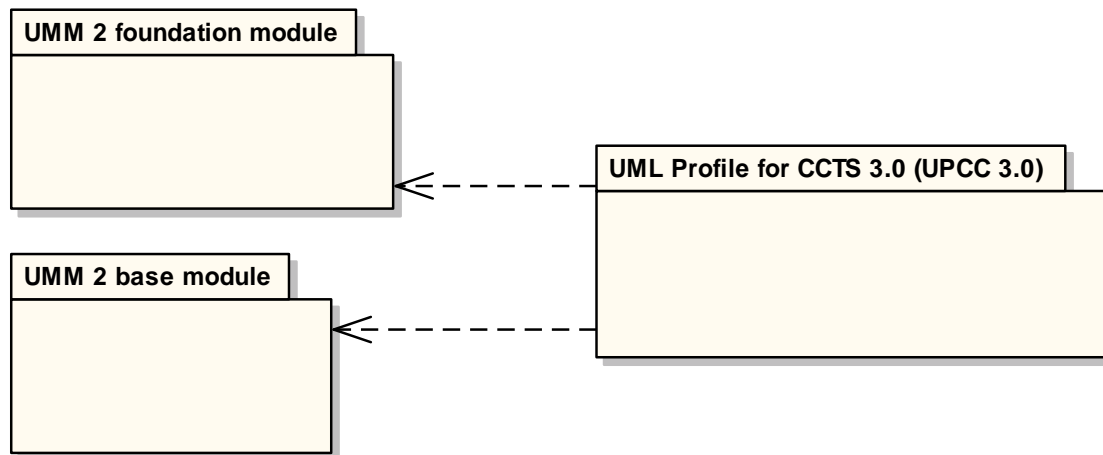
- If you want to learn modeling, it is not a good idea to start with the meta model
- If you are an experienced modeler the meta model serves as reference
- A tool builder must implement the meta model to provide the modeling environment for the modeler

- Major shortcoming of Core Components
 - missing formalized representation model
 - no direct integration into modeling tools possible
- UPCC goals
 - Define UML Profile for CCTS to allow for an unambiguous representation of Core Components in UML
 - Support validation of structure and semantics of CCTS compliant information models
 - Provide an unambiguous basis for the derivation of XML Schema artifacts
 - Make CCTS information modeling available to a broad user community
 - Help to improve model interchange between UML tools of different vendors
- Version
 - Version 1.0 (official UN/CEFACT standard) – compliant to CCTS 2.01
 - Version 3.0 (under development) – compliant to CCTS 3.0
 - Editor: Philipp Liegl, Vienna University of Technology (Austria)

The UML Profile approach

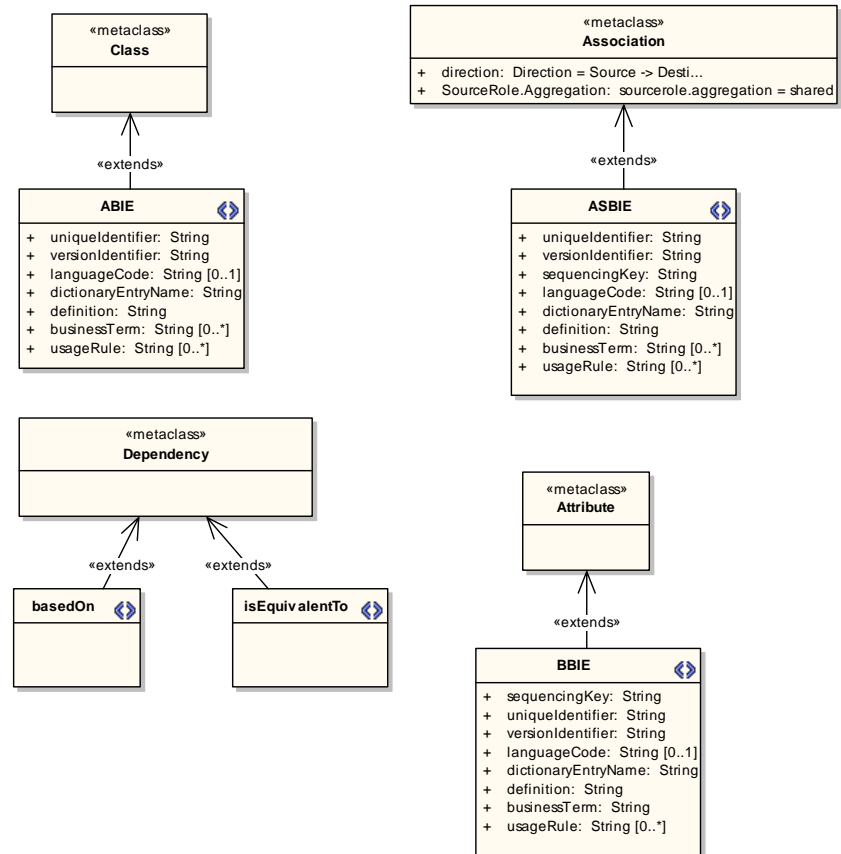


Dependency between UN/CEFACT's Modeling Methodology (UMM) and the UML Profile for Core Components (UPCC)

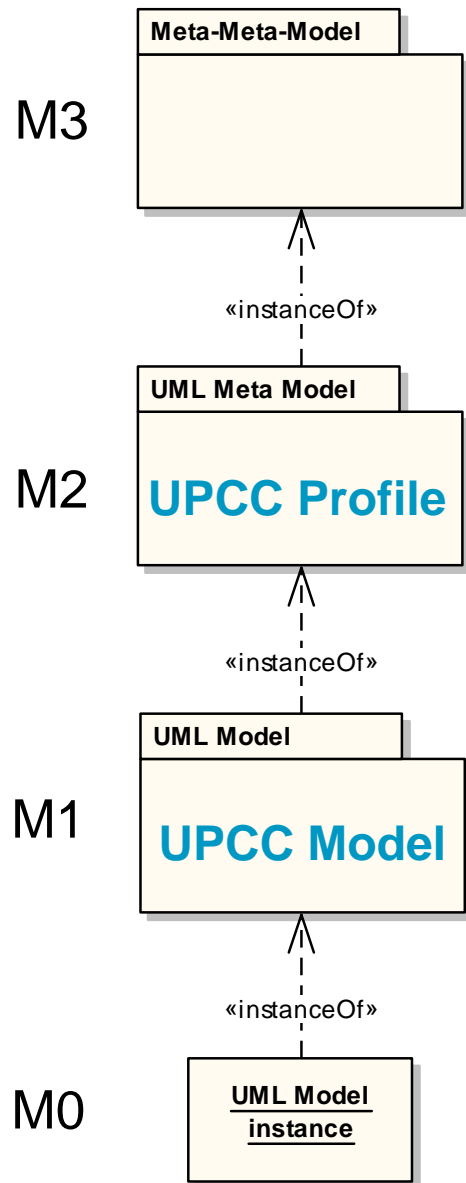


The idea of a UML Profile

- Provide a generic extension mechanism for customizing the UML meta model to a particular application domain
- UML Profile comprises three distinctive parts
 - Stereotypes
 - Tagged Values
 - Constraints



UML Meta Model?



UML Infrastructure – Definition of the elements that can be used to define the different UML model types

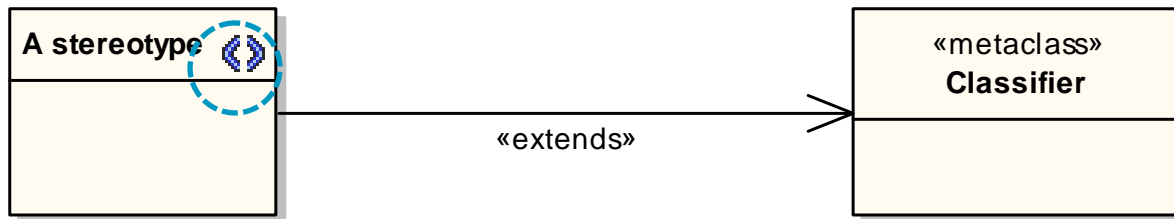
UML Superstructure – Definition of the different UML model types

Model of a specific System

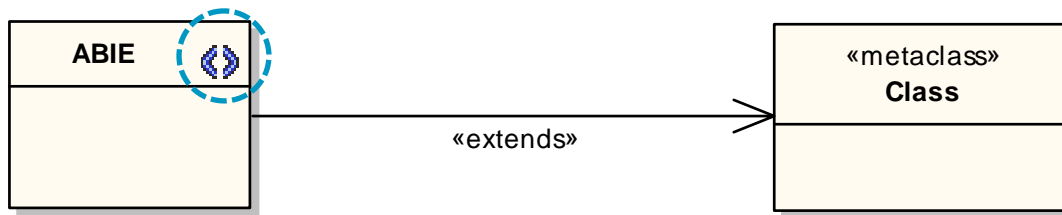
Instance of the specific model

The idea of a UML Profile cont'd - stereotypes

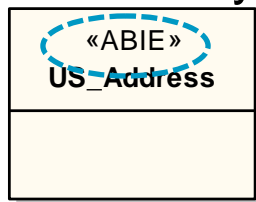
- Stereotypes are defined by extending a metaclass from the UML meta model




- UPCC meta model example



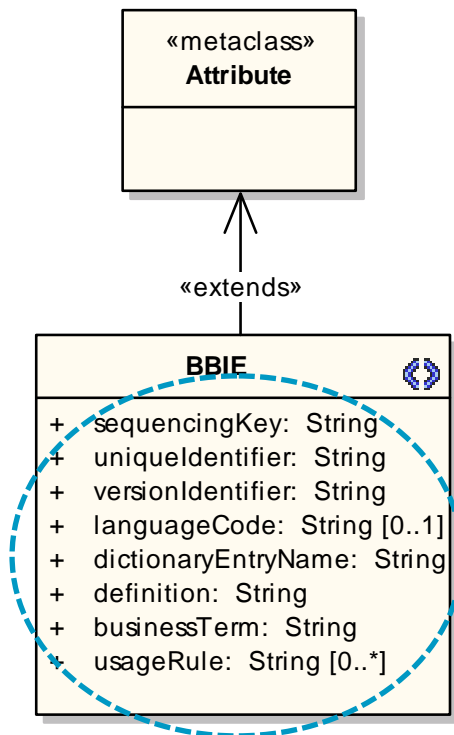
- ABIE stereotype used in a concrete UPCC instance



 indicates a stereotype

The idea of a UML Profile cont'd – tagged values

- Tagged values are defined as part of a stereotype
- Define a stereotype in more detail
- Tagged Value: always a Key-Value pair
- Example:
 - A basic core component has a dictionary entry name, a definition etc.



 tagged values

The idea of a UML Profile cont'd – constraints

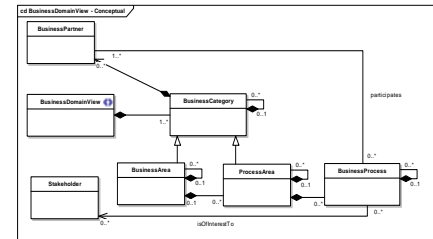
- Provide additional restrictions on the usage of certain stereotypes and their interdependencies
- In the UPCC specification constraints are specified in two representative forms
 - Free form text
 - Object Constraint Language Specification (OCL)
- Example:
 - An ABIE shall contain only BBIEs and ASBIEs. An ABIE shall contain at least one BBIE or ASBIE.
 - OCL:
`context Class inv: self.ABIE() ...`

Definition of each section in the UPCC 3.0 specification

1. Abbreviations of stereotypes

Stereotype Abbreviation	Full Stereotype Name
bDomainV	BusinessDomainView
bCategory	BusinessCategory
bArea	BusinessArea
ProcessArea	ProcessArea
bProcessUC	BusinessProcessUseCase
bProcess	BusinessProcess
bProcessAction	BusinessProcessAction
bESharedState	SharedBusinessEntityState
bEInternalState	InternalBusinessEntityState

2. Conceptual Description (informative)

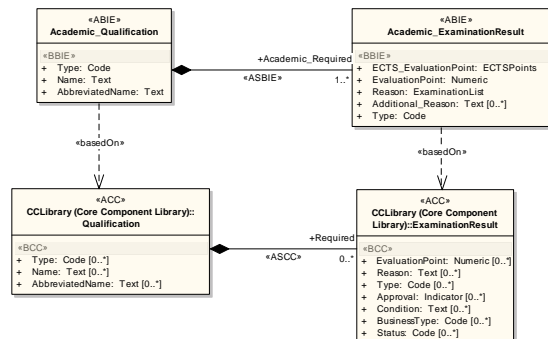


3. Stereotypes and Tag Definitions (normative)

Stereotype	BusinessPartner
Base Class	Actor
Parent	Stakeholder
Description	A business partner is an organization type, an organizational unit type or a person type that participates in a business process. Business partners typically provide input to and/or receive output from a business process. Due to the fact that a business partner participates in a business process she or he has by default a vested interest in the business process. It follows that a business partner is a special kind of stakeholder.
Tag Definition	Inherited tagged values: - interest

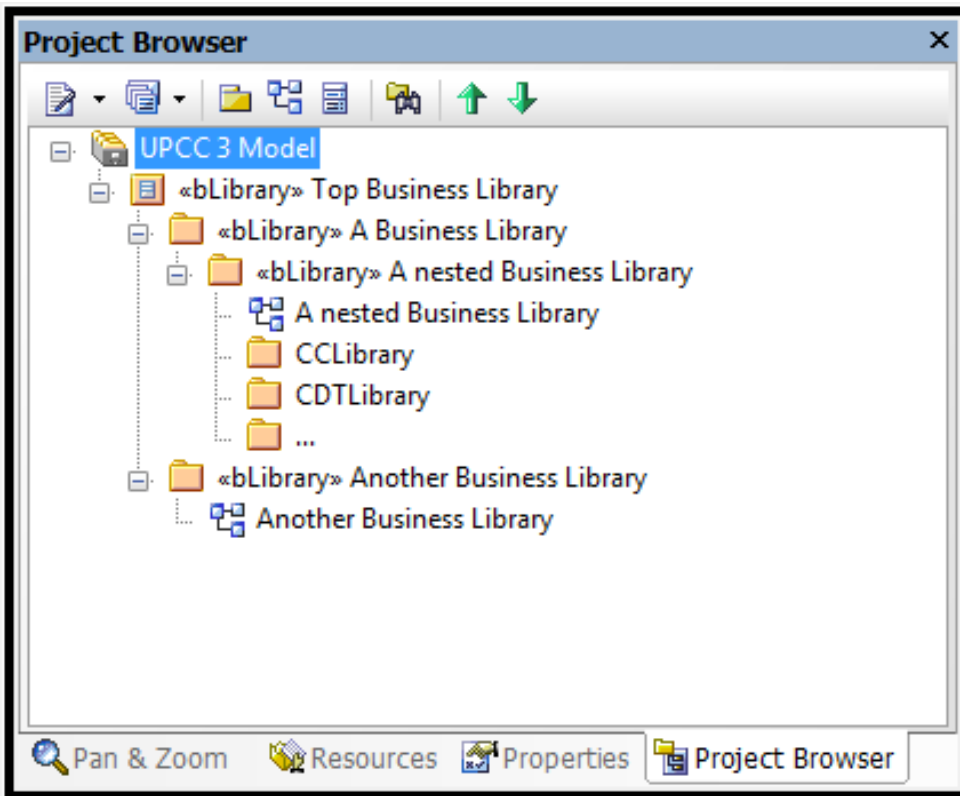
4. Constraints (normative)

5. Example

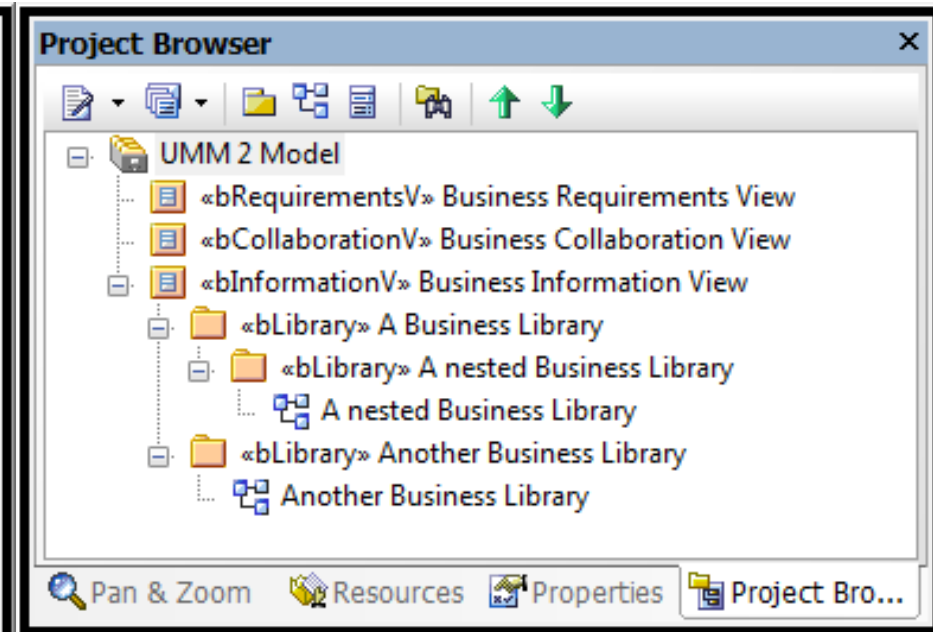


UPCC in Enterprise Architect

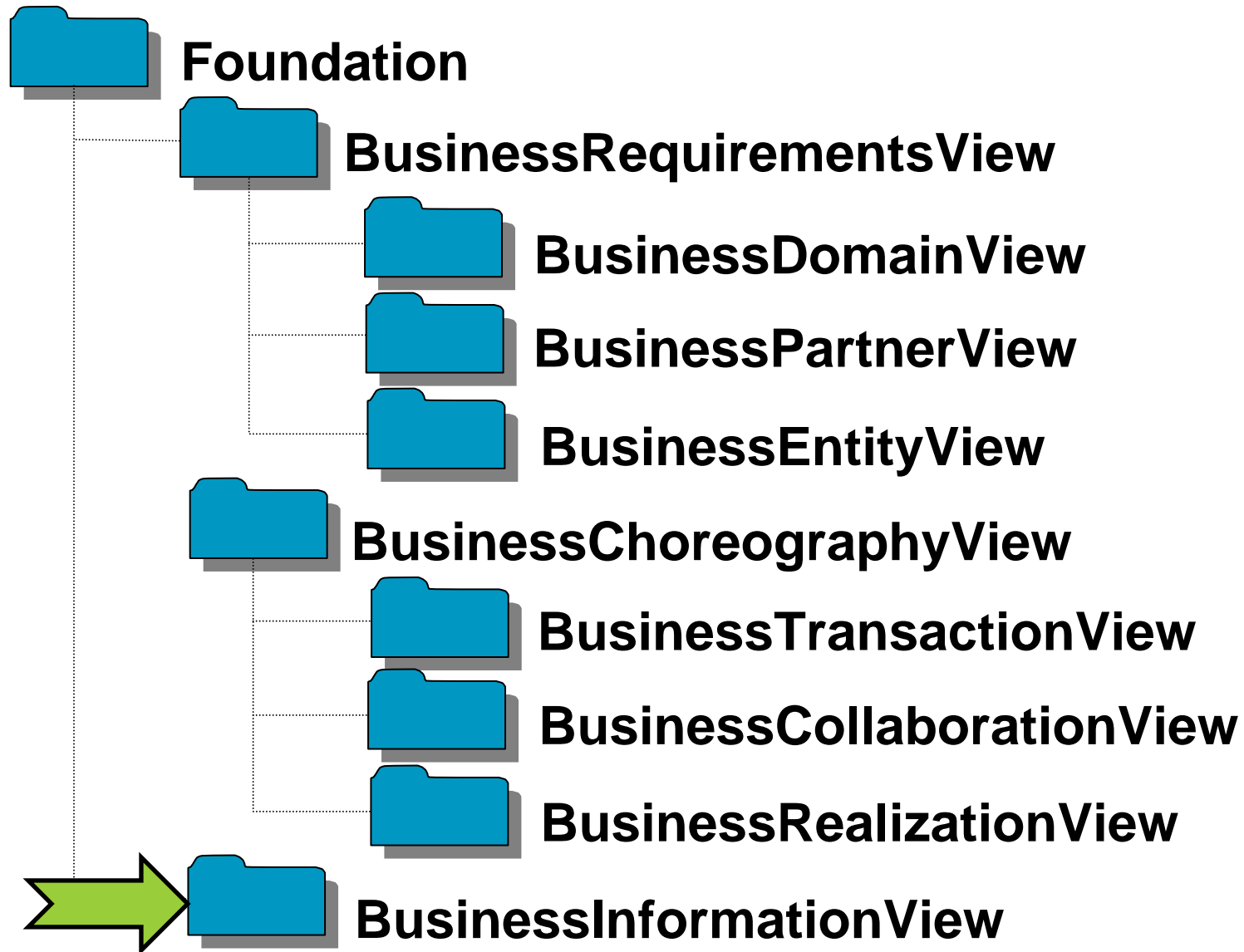
Standalone data model



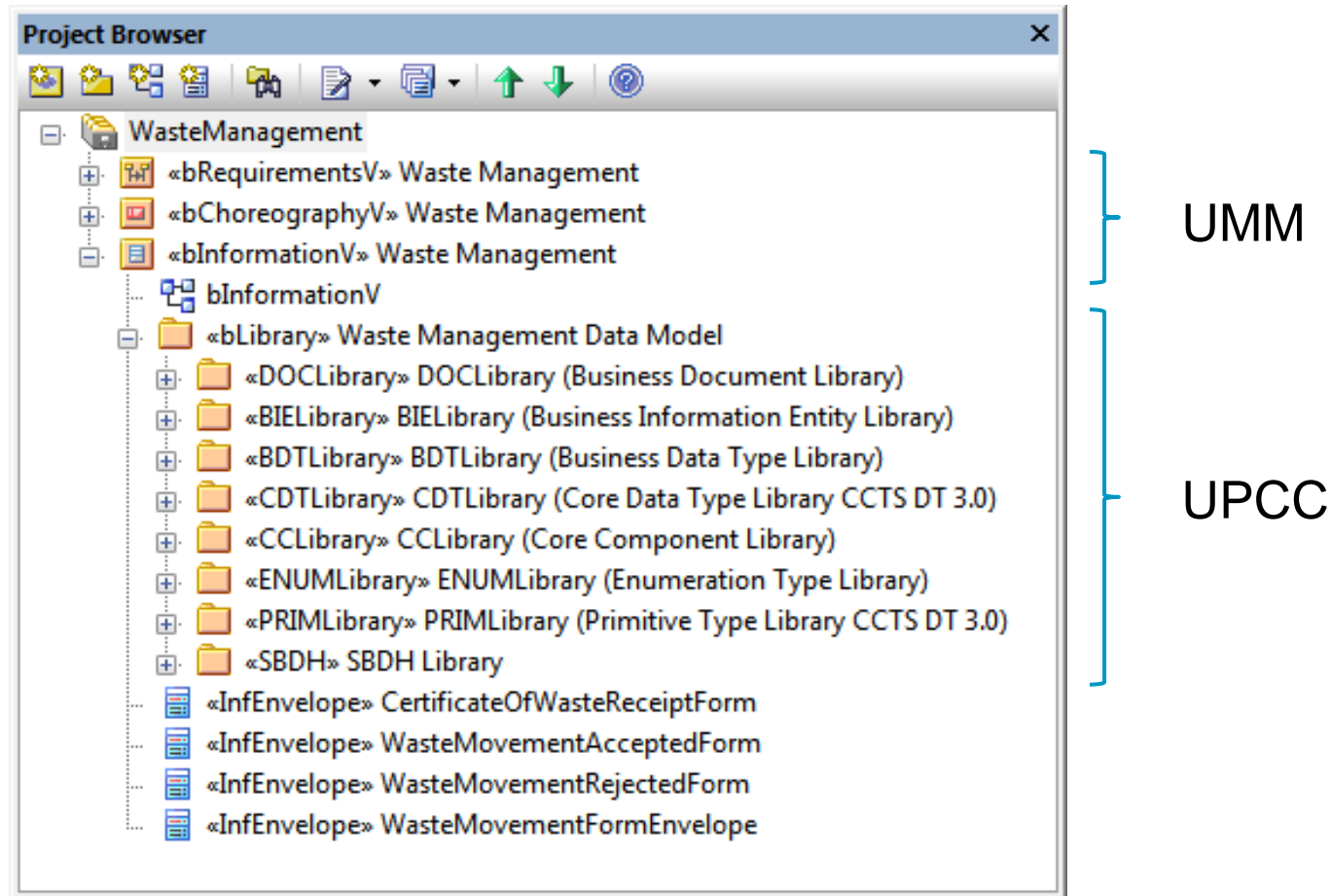
Integration in a UMM 2 model



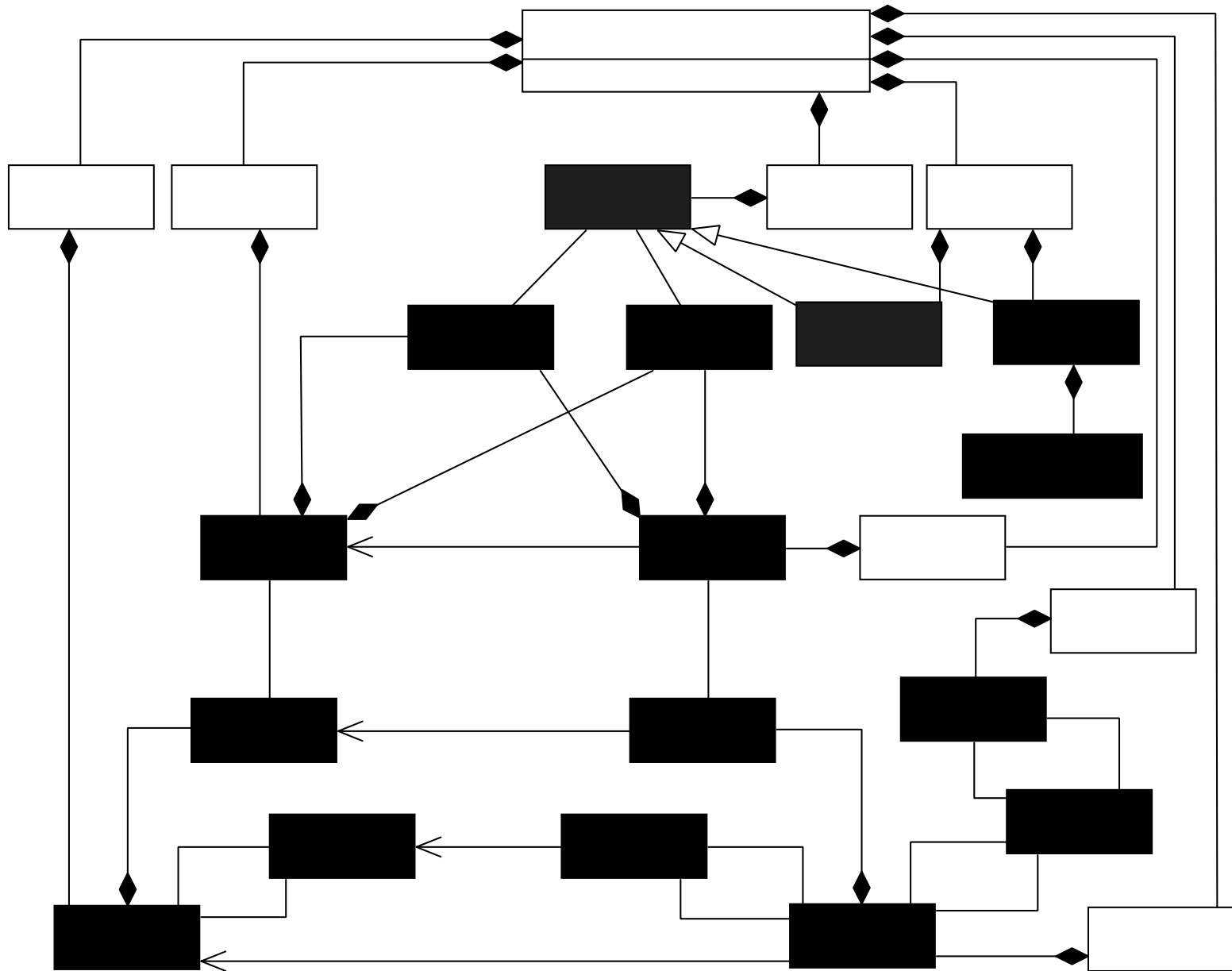
Integration of UPCC in UN/CEFACT's Modeling Methodology (UMM)



Integration of UMM and UPCC cont'd rsa

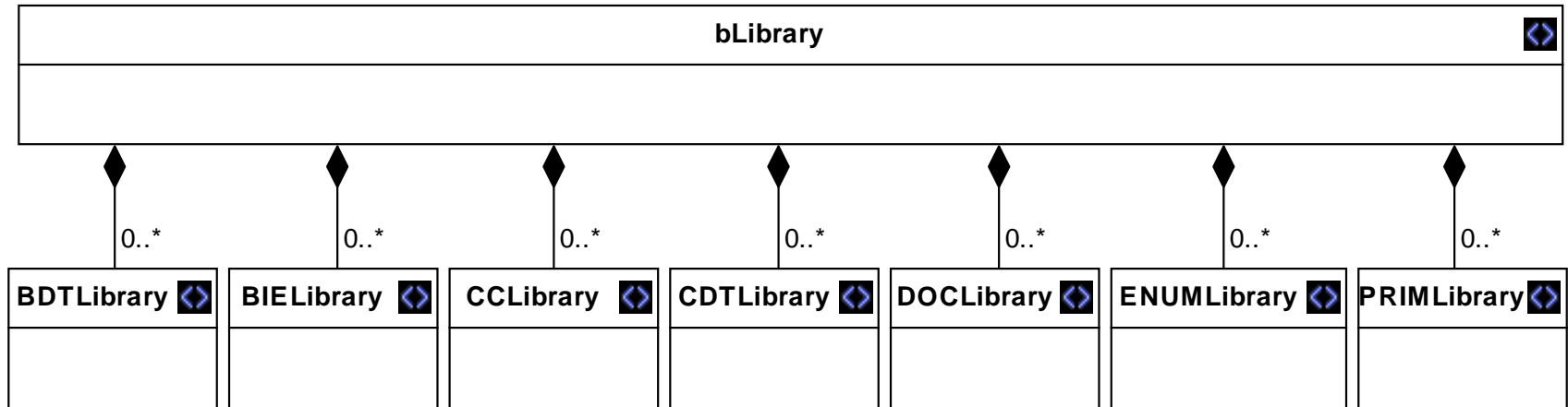


UPCC – the big picture



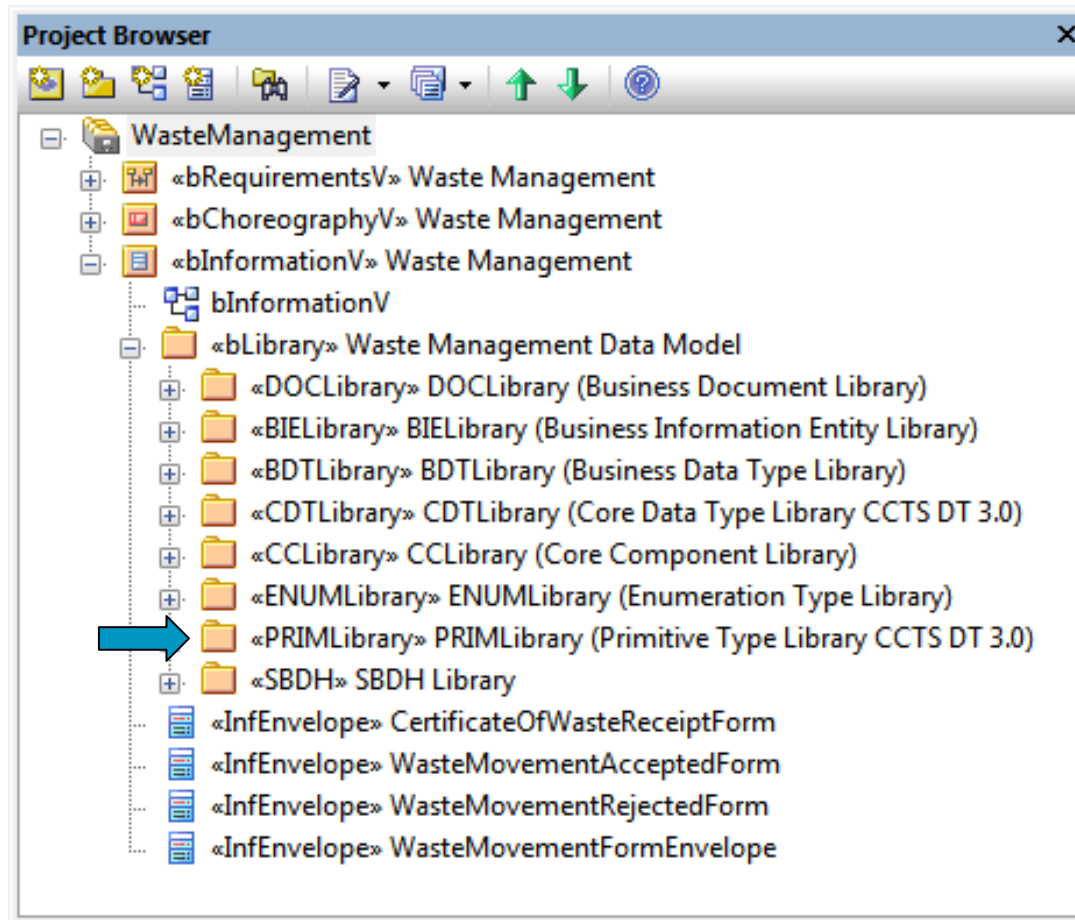
UPCC package overview

from UMM 2.0
Base Module



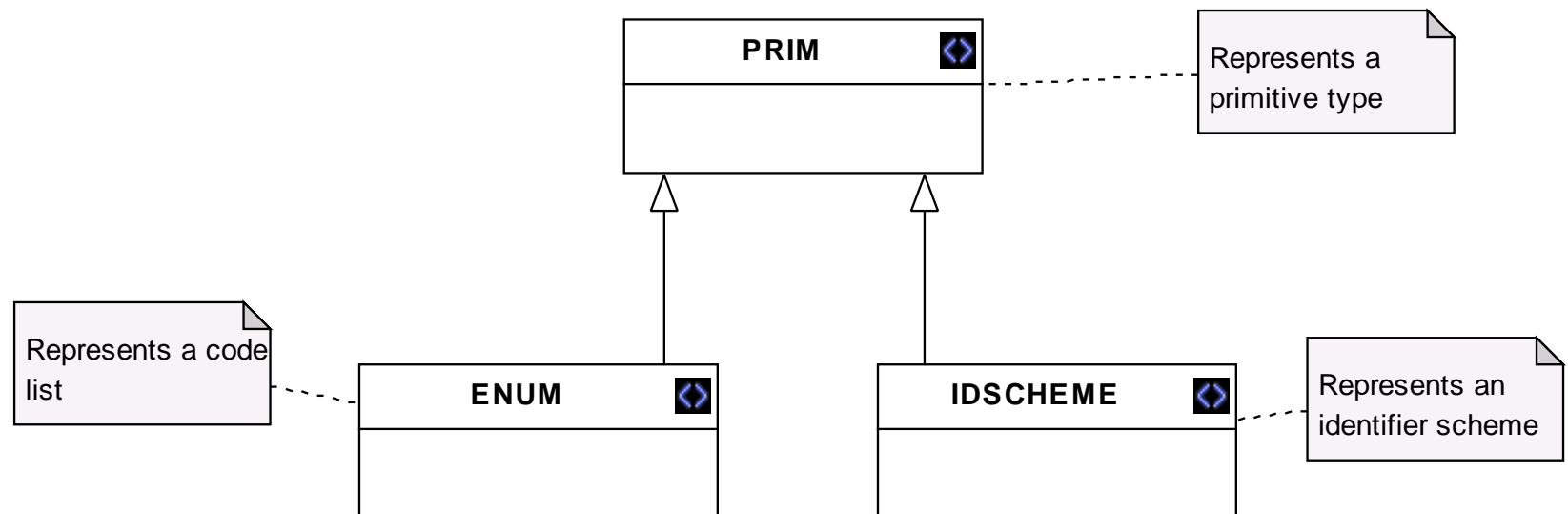
- For each artifact type a dedicated library is assigned
 - BDTLibrary Business Data Type Library
 - BIELibrary Business Information Entity Library
 - CCLibrary Core Component Library
 - CDTLibrary Core Data Type Library
 - DOCLibrary Business Document Library
 - ENUMLibrary Enumeration Library
 - PRIMLibrary Primitive Type Library

PRIMLibrary - Primitive Type Library

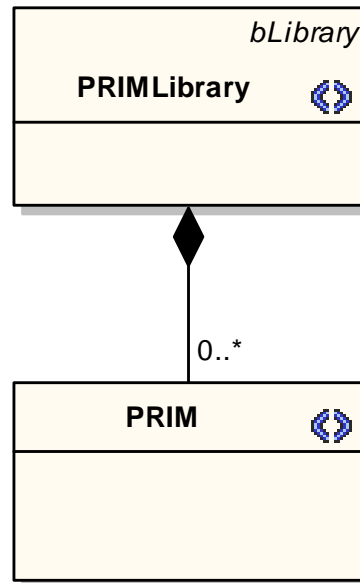


PRIMLibrary - Primitive Type Library

- A primitive type library is used to aggregate **primitive types (PRIM)**
- Primitives are used to set the value domains of **content components (CON)** and **supplementary components (SUP)** of **Core Data Types (CDT)** and **Business Data Types (BDT)**
- A primitive type has two specializations:
 - Enumeration type (ENUM) – represents a code list
 - Identifier scheme type (IDScheme) – represents an identifier scheme



PRIMLibrary – Primitive Type Library (conceptual)



PRIM Library – according to CCTS DT Catalogue 3.0

«PRIM» Binary

«PRIM» Float

«PRIM» NormalizedString

«PRIM» Boolean

«PRIM» Integer

«PRIM» String

«PRIM» Decimal

«PRIM» Double

«PRIM» TimeDuration

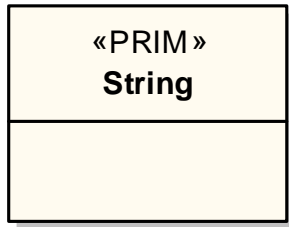
«PRIM» TimePoint

«PRIM» Token

Restricting Primitive Types

- Primitive Types may be restricted using the concept of facets
- Allowed facets per primitive type are specified in the CDT DT Catalogue 3.0
- e.g **String**: allowed facets:
 - [Enumeration](#)
 - [Length](#)
 - [Minimum Length](#)
 - [Maximum Length](#)
 - [Pattern](#)
- In UPCC facets per primitive type are specified using constraints

Restriction of a Primitive Type



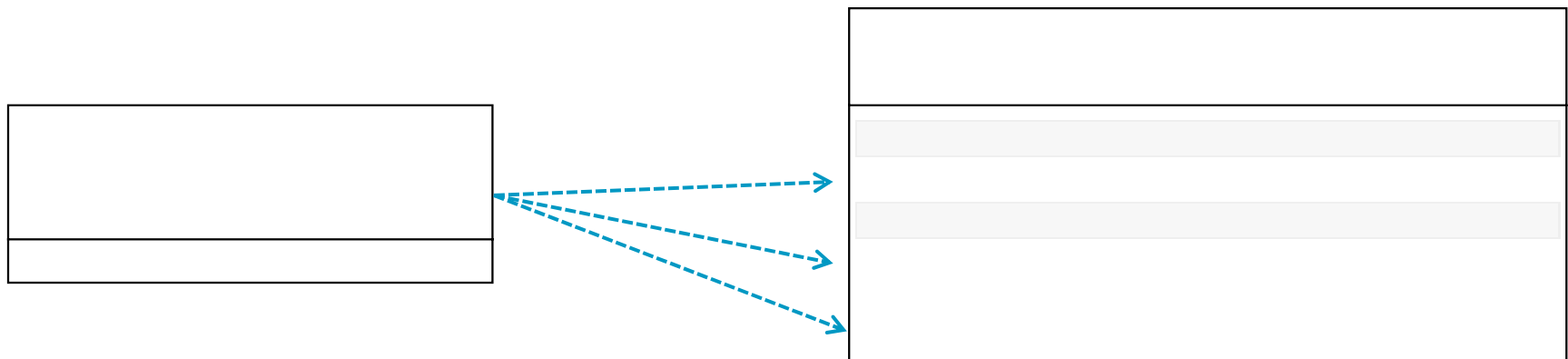
Tagged Values

String (Class)

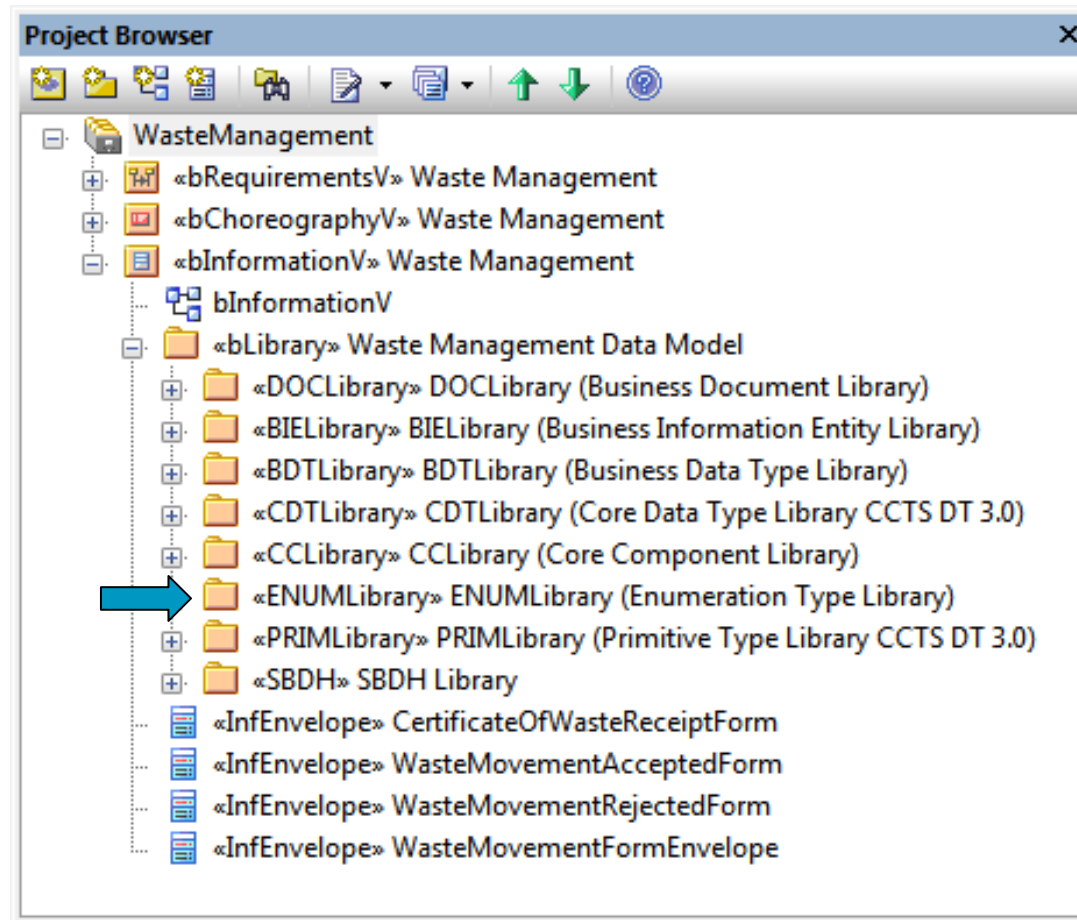
businessTerm	
definition	
dictionaryEntryName	String
fractionDigits	
languageCode	
length	
maxExclusive	
maxInclusive	
maxLength	5
minExclusive	
minInclusive	
minLength	2
pattern	
totalDigits	
uniqueIdentifier	C6BF07AA-AE2A-4808-B117-C444E54ED064
versionIdentifier	

Notes Tagged V... Pan & Zo... Properties Model Vi... Resources

Setting the value domain of content components and supplementary components - example

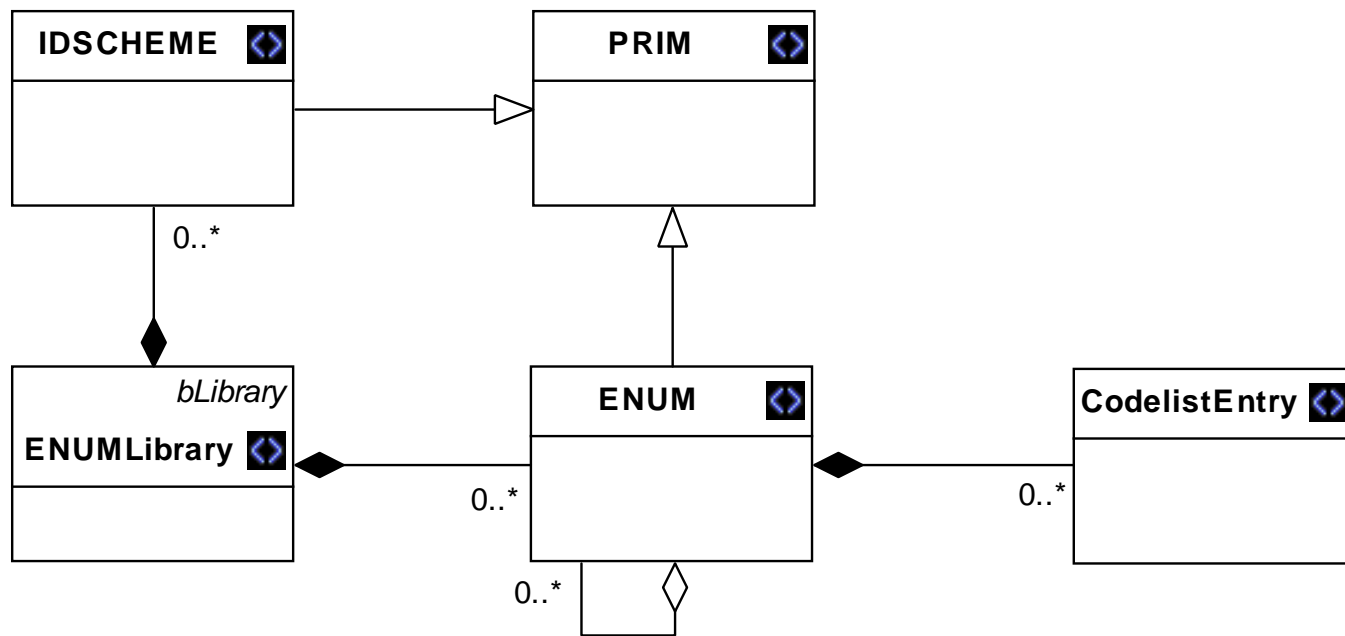


ENUMLibrary – Enumeration Type Library



ENUMLibrary – Enumeration type library

- An enumeration library is used to aggregate
 - Identifier schemes (IDSCHEME)
 - Code lists (ENUM)



Enumeration Type Example

- Enumeration types (ENUM) are used to depict code lists

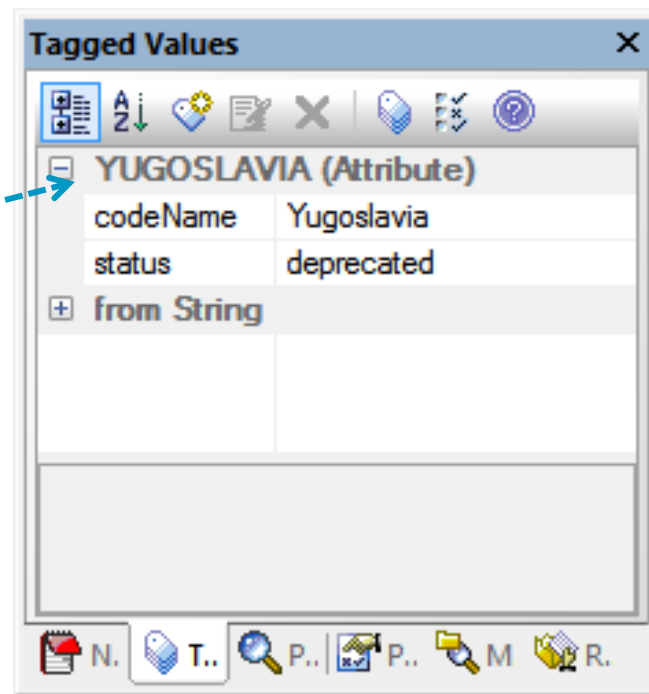
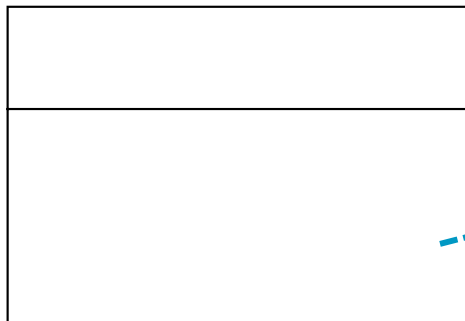
«ENUM» ISO42173A	
+ AED = United Arab Emi...	
+ AFN = Afghani	
+ CHF = Swiss franc	
+ ... = ...	



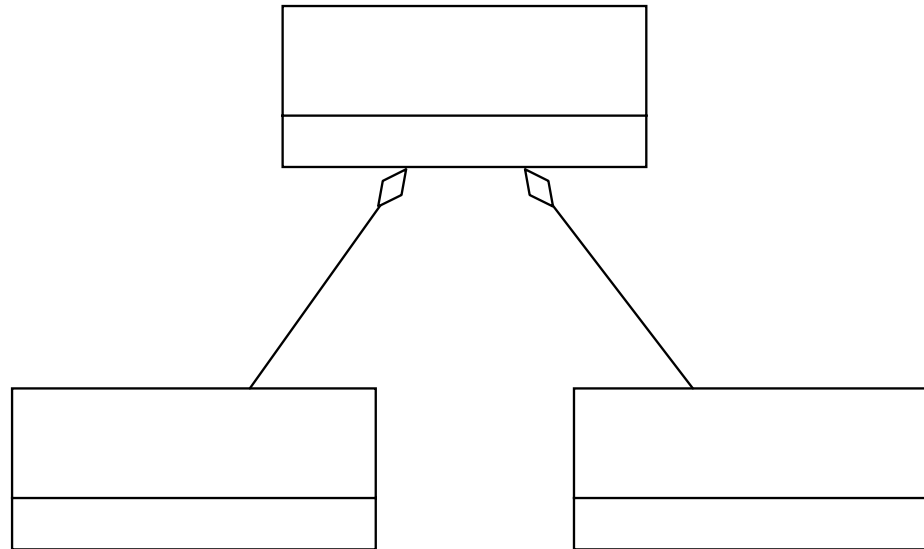
Tagged Values	
ISO42173A (Class)	
agencyIdentifier	5
agencyName	ISO
allowedPrimitives	String
businessTerm	-
codeListIdentifier	ISO42173A
codeListName	Codes for the representation of currencies...
coreValueDomainDefault...	true
dictionaryEntryName	ISO4217A
enumerationURI	-
languageCode	en-GB
modificationAllowedIndic...	true
uniqueIdentifier	3252220C-2E58-11DE-A6EB-C57056D89...
versionIdentifier	1.0

Enumeration values

- An enumeration consists of an enumerated set of values (**CodeListEntry**)
- **CodeListEntry** carries additional meta-information about the code list entry

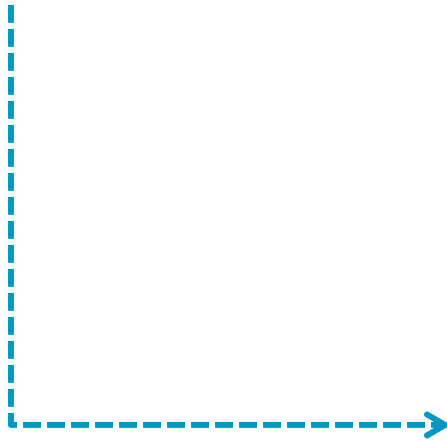


Combining code lists to a new code list



Identifier scheme example

- Identifier schemes (IDSCHEME) are used to depict non-enumerated **identifier scheme values**



Tagged Values

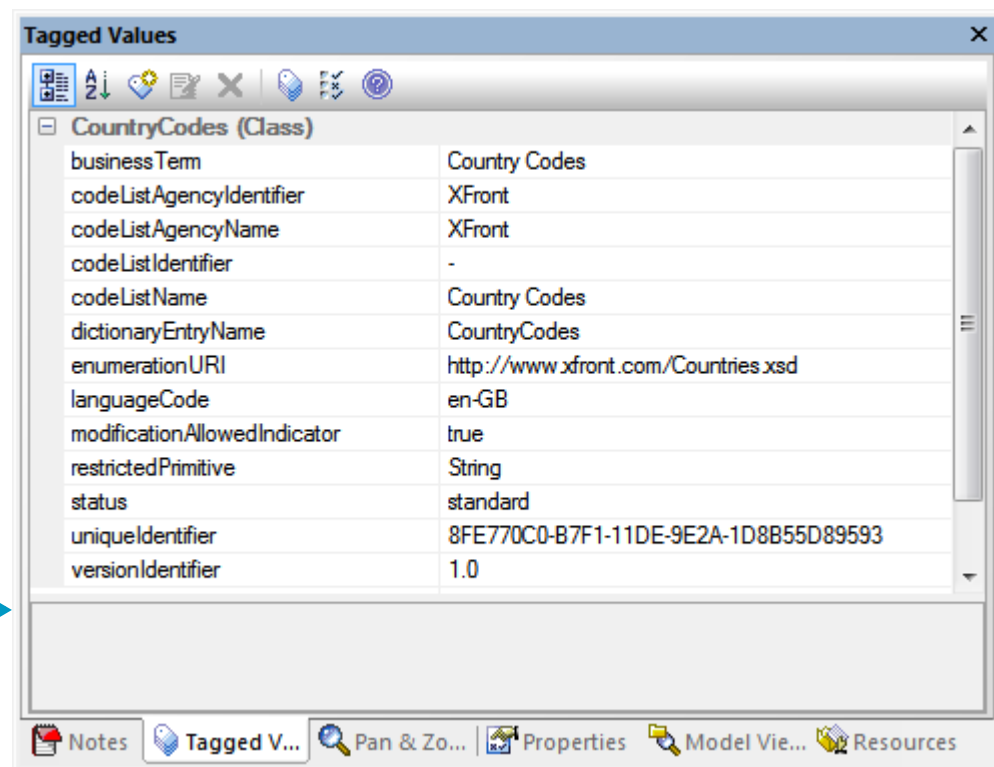
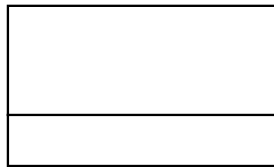
Australian_books_and_serials (Class)

businessTerm	Australian books and serials identifier
definition	An identifier scheme for Australian books and ...
dictionaryEntryName	Australian_books_and_serials
identifierSchemeAgencyIdentifier	NLA
identifierSchemeAgencyName	National Library of Australia
pattern	nla.(gen aus)-(issn\d{8}[a-z]? (an vn)\d{1,12}[...
restrictedPrimitive	String
uniqueIdentifier	6311799C-B80A-11DE-92E4-174D56D89593
versionIdentifier	1.0

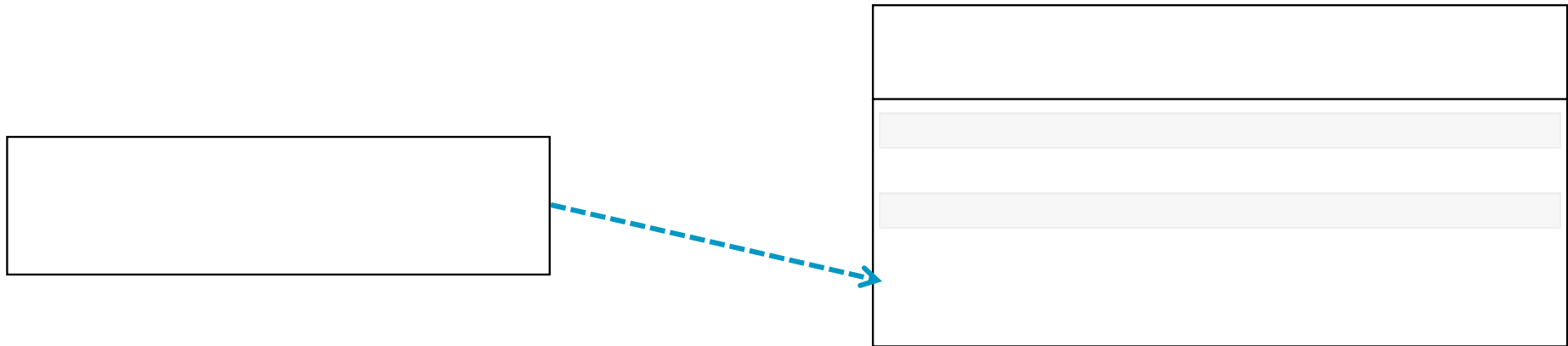
Notes Tagged V... Pan & Zo... Properties Model Vi... Resources

Reusing existing identifier schemes or code lists

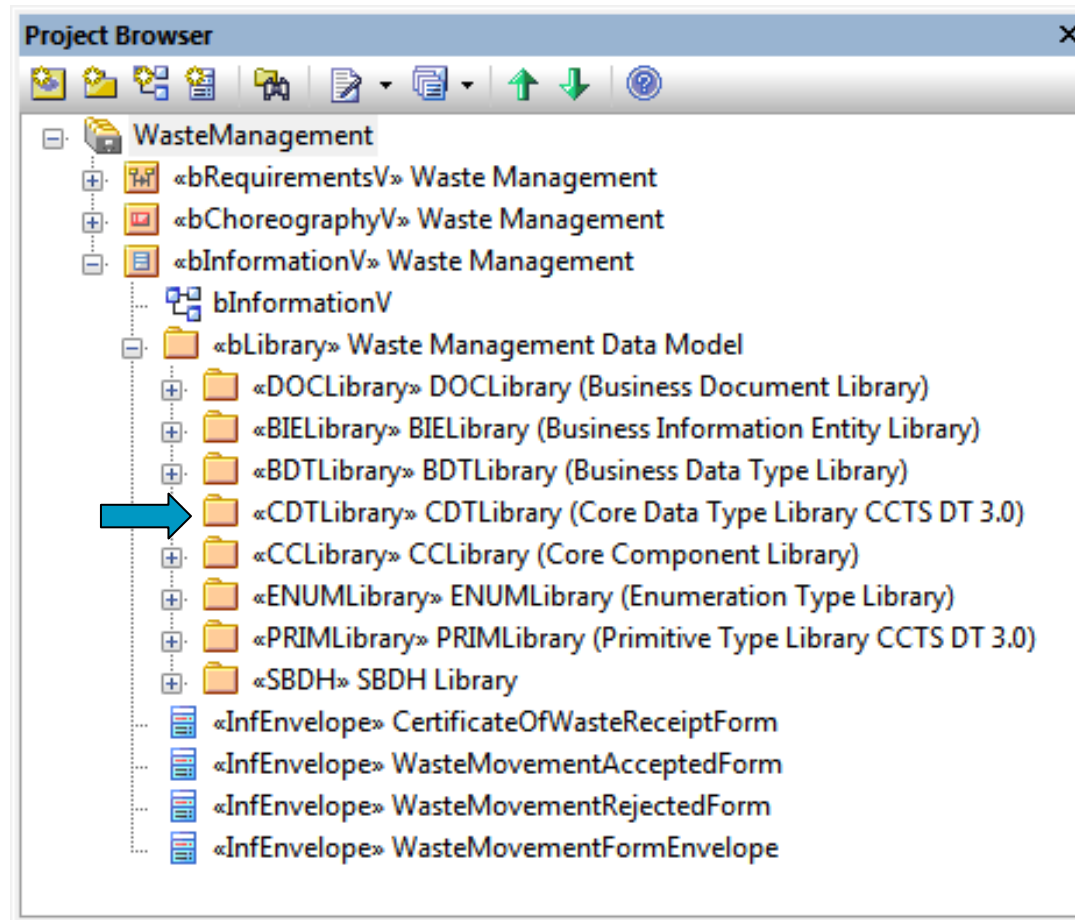
- **Solution A:** Import identifier scheme or code list directly into the UML tool
 - Prerequisite: Scheme or lists must be provided in pertinent XML format
- **Solution B:** Point to an existing identifier scheme or code lists using a URI



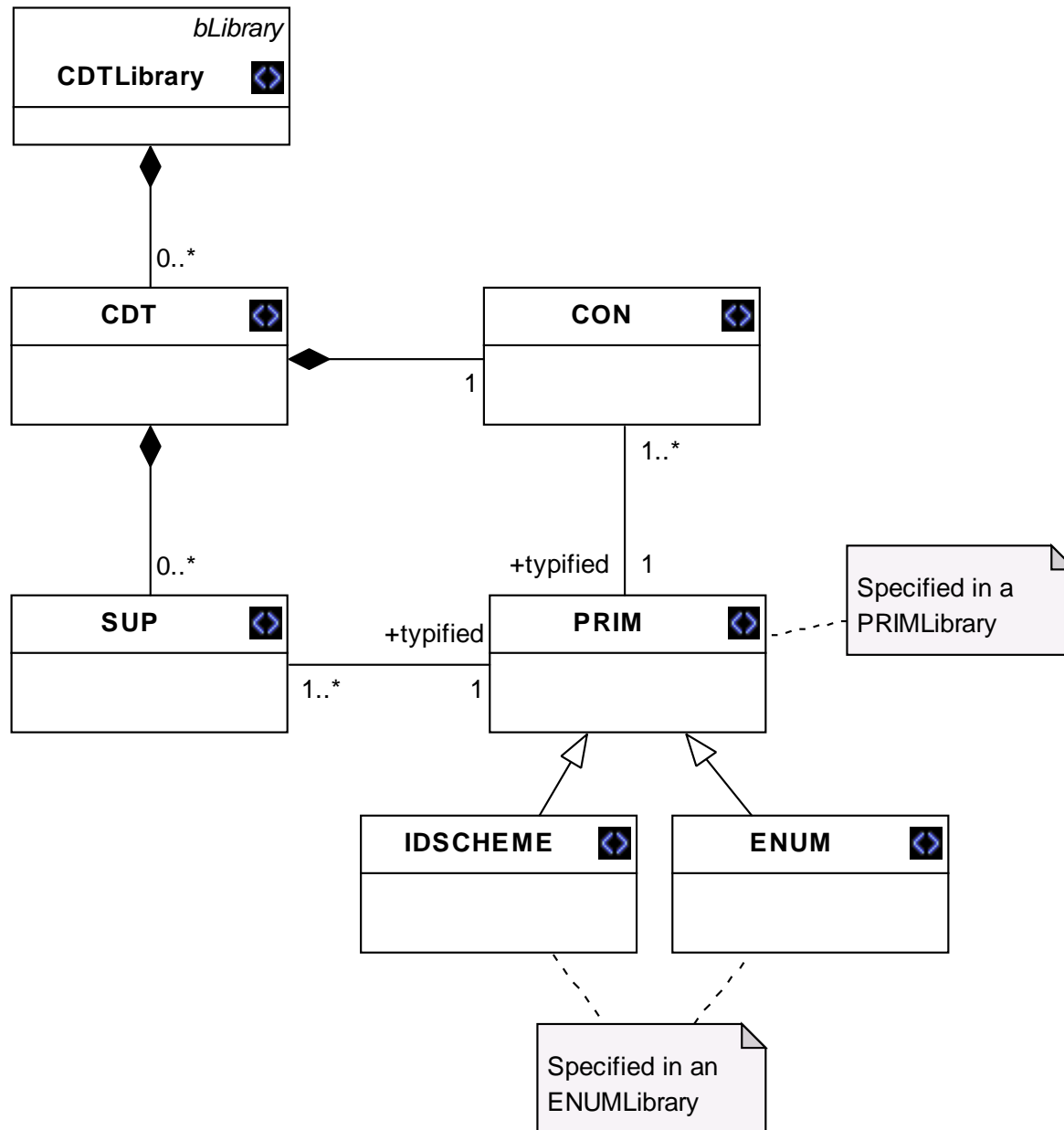
Setting the value domain of content components and supplementary components



CDTLibrary – Core Data Type Library



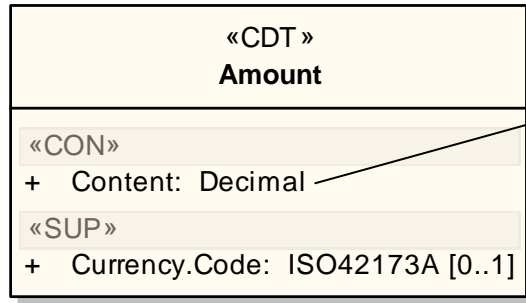
CDTLibrary - Core Data Type Library (conceptual)



Core Data Types revisited

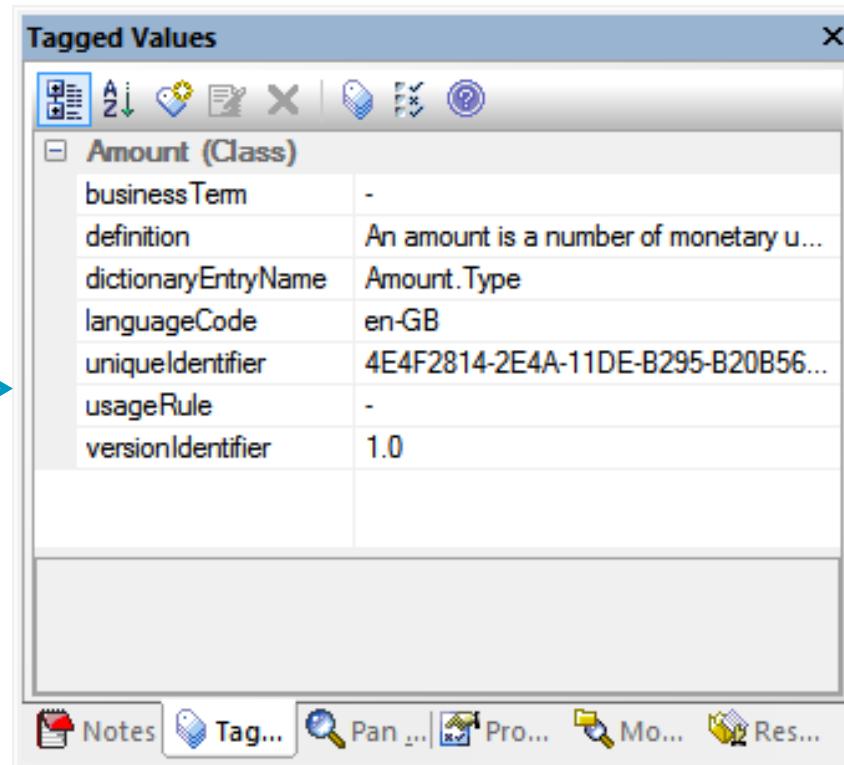
- Core Data Types are used to set the **value domain** of CCTS 3.0 Basic Core Components Properties
- Core Data Types are the **basis for** all CCTS 3.0 conformant **Business Data Types (BDT)**
- UN/CEFACT releases a set of allowed Core Data Types in the UN/CEFACT Data Type Catalogue
- All CDT definitions **must comply** to the Data Type Catalogue
- No new CDTs may be created, otherwise compatibility to the UN/CEFACT Core Component Library is lost!

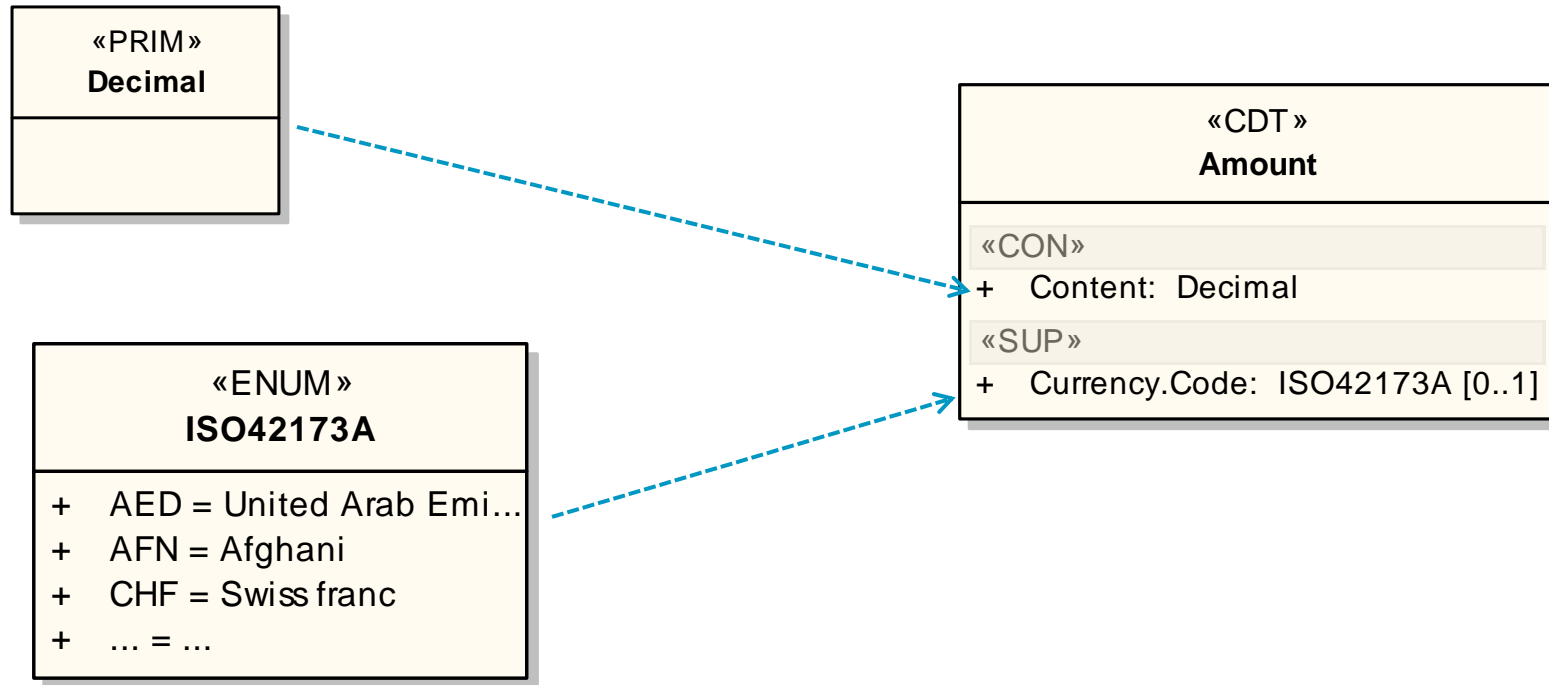
CDT Library – example according to CCTS DT Catalogue 3.0



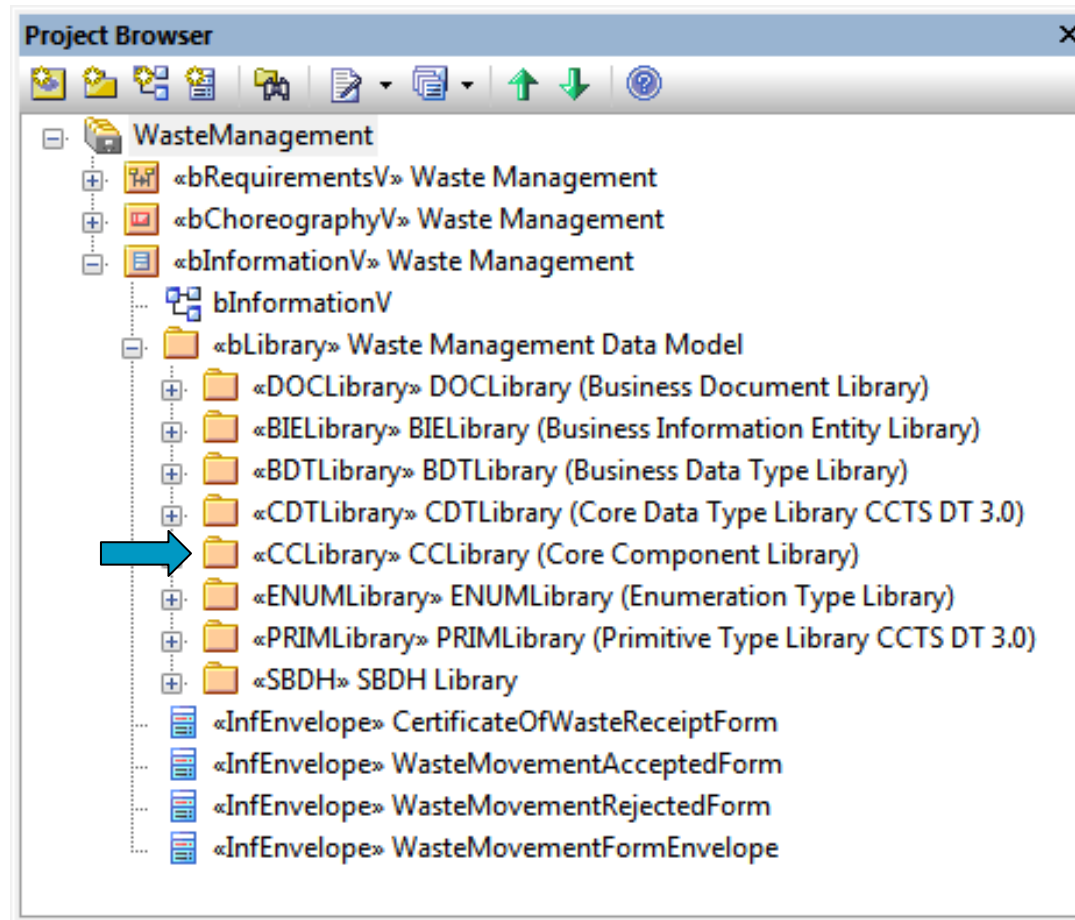
Additionally allowed primitives:

Decimal
Double
Float
Integer

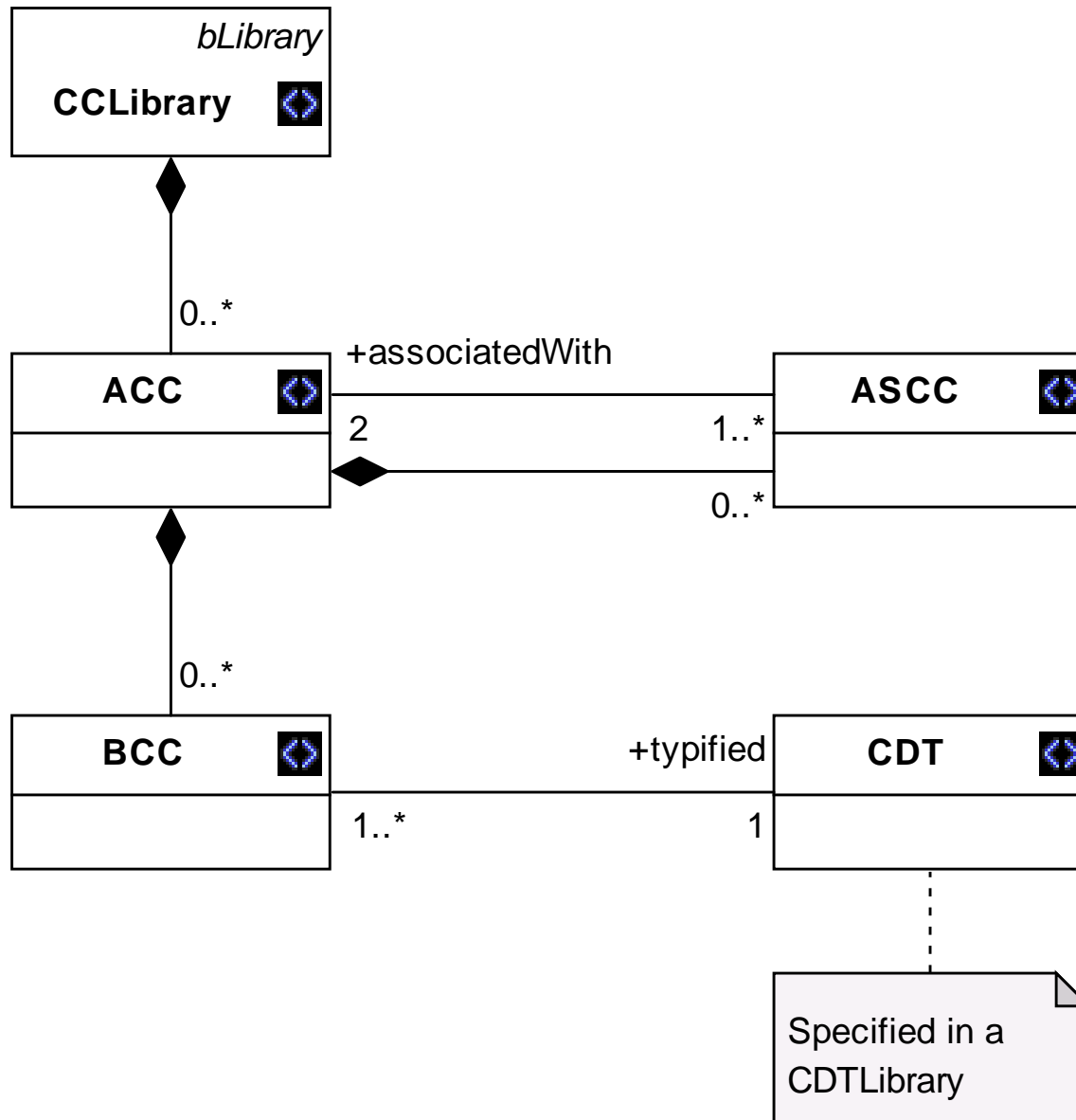




CCLibrary – Core Component Library

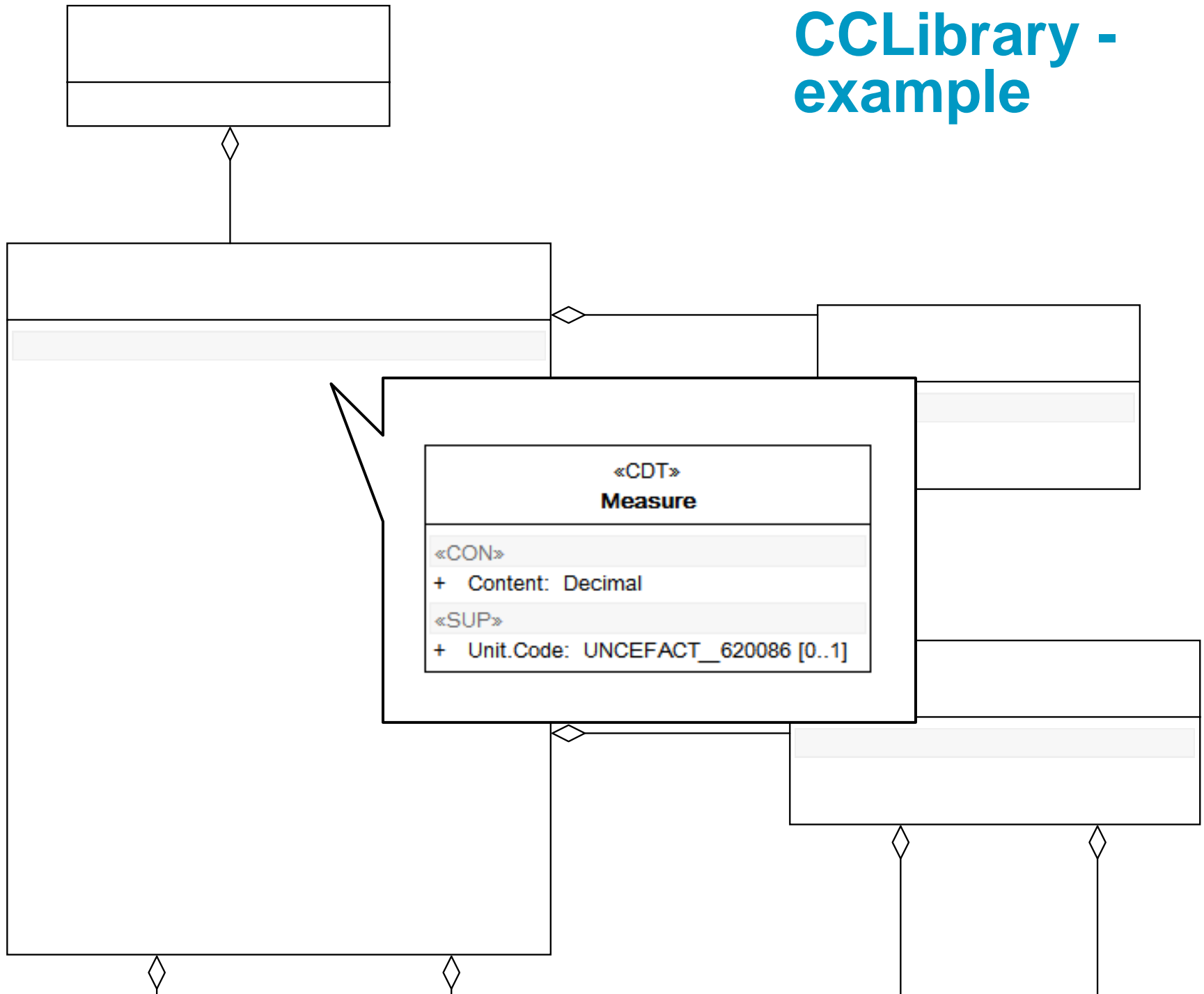


CCLibrary – Core Component Library (conceptual)

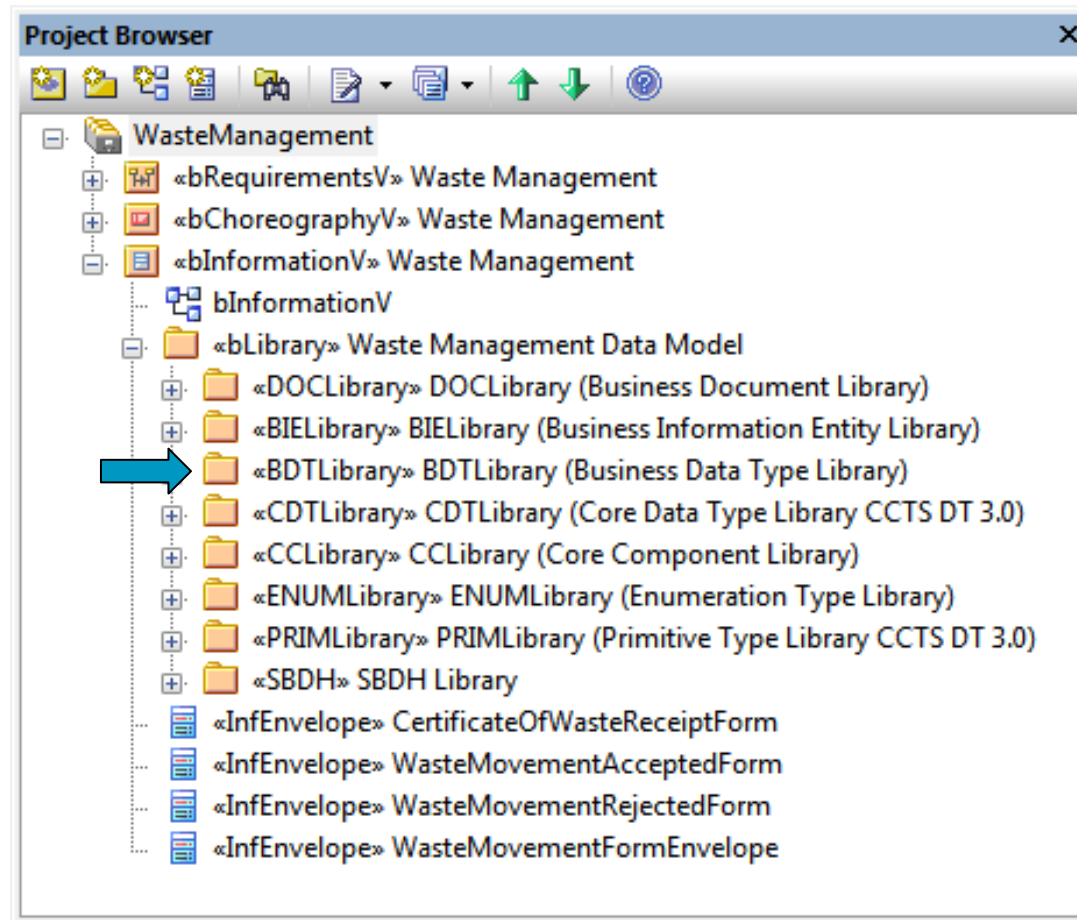


- A Core Component Library is a **UML representation of the UN/CEFACT Core Component Library**
- A business document modeler **must not create new Core Components**, but reuses existing Core Components from the UN/CEFACT Library
- If user-specific Core Components are created, compatibility to the UN/CEFACT Library is lost
 - Might be considered in case only partner specific compatibility is required
- Usual workflow: Use Core Component from the CCLibrary and create new **Business Information Entities**

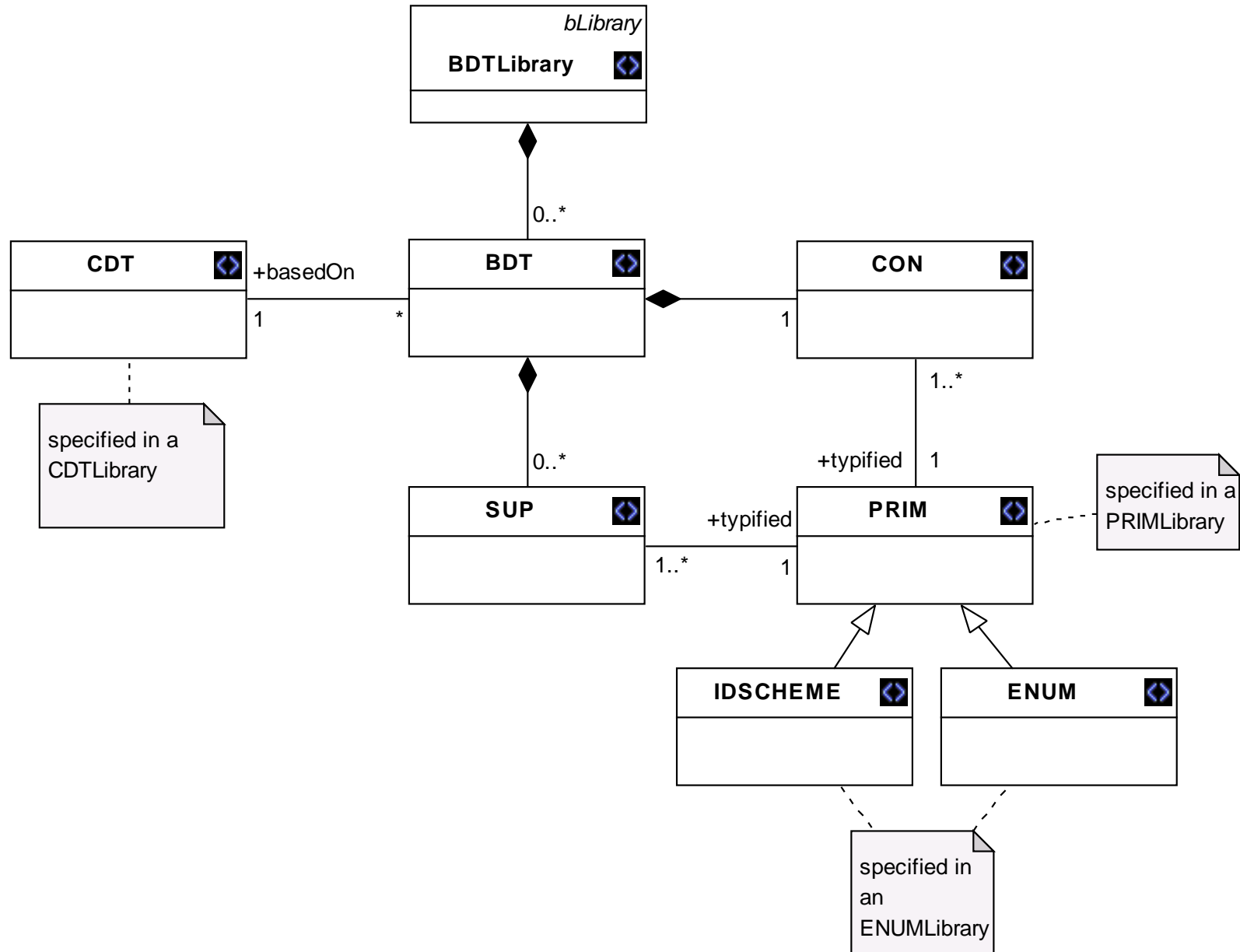
CCLibrary - example



BDTLibrary – Business Data Type Library

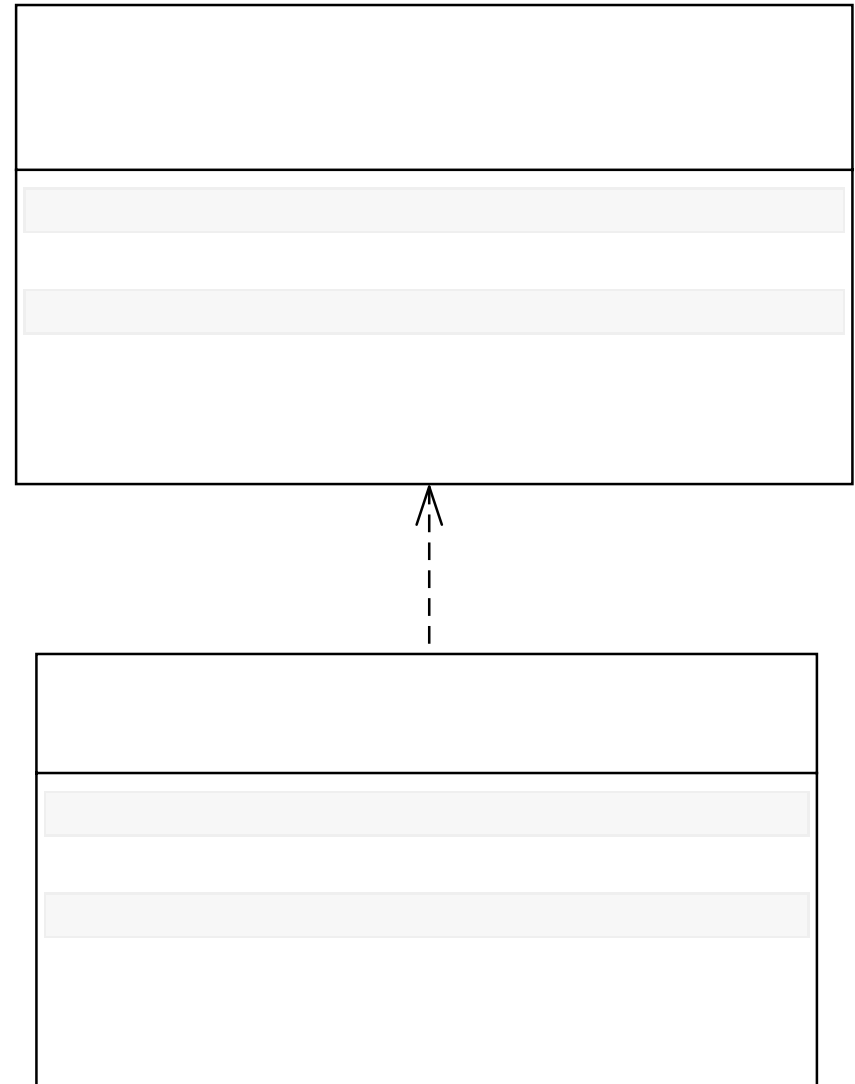


BDTLibrary – Business Data Type Library (conceptual)

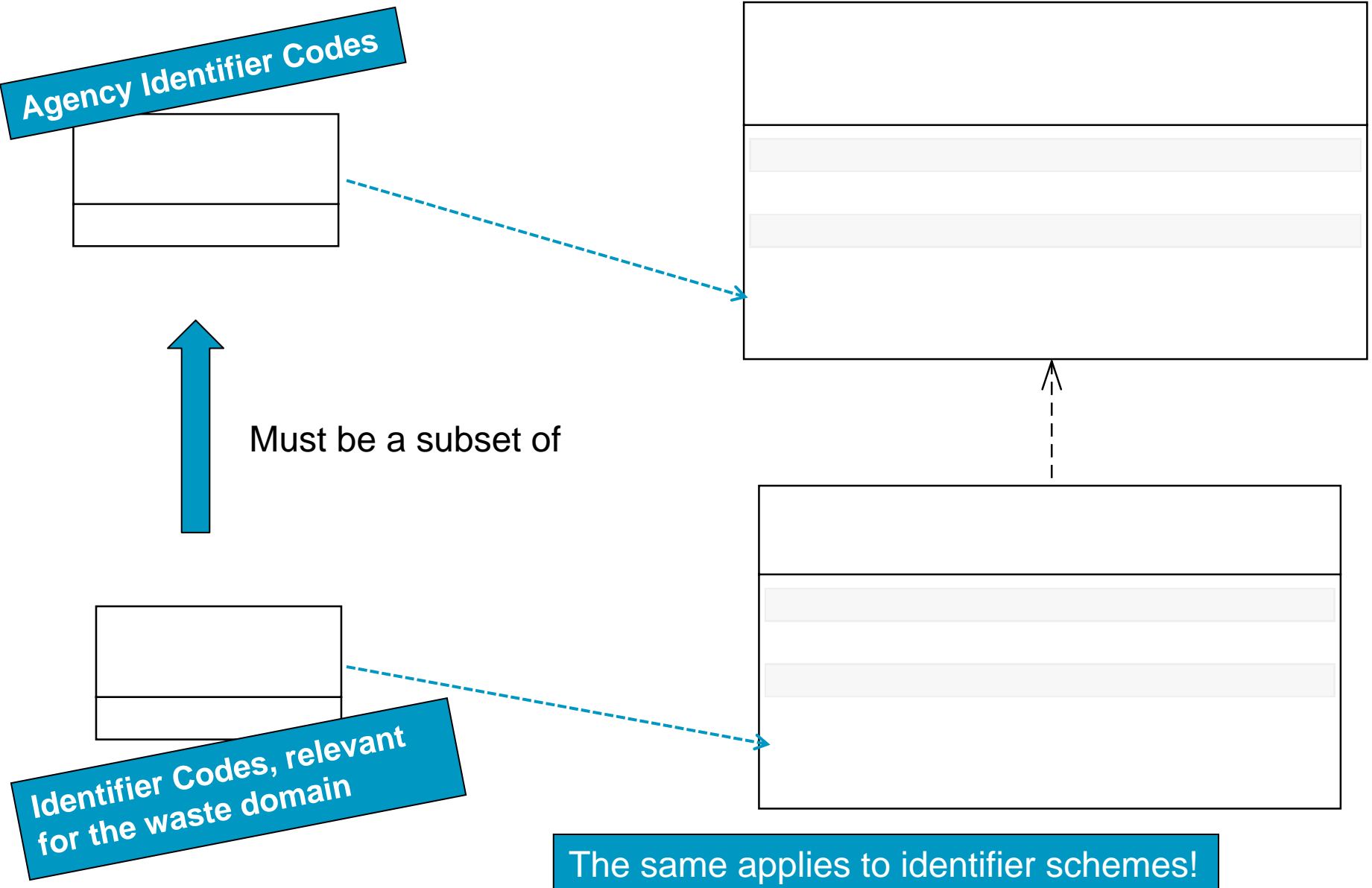


Business Data Types

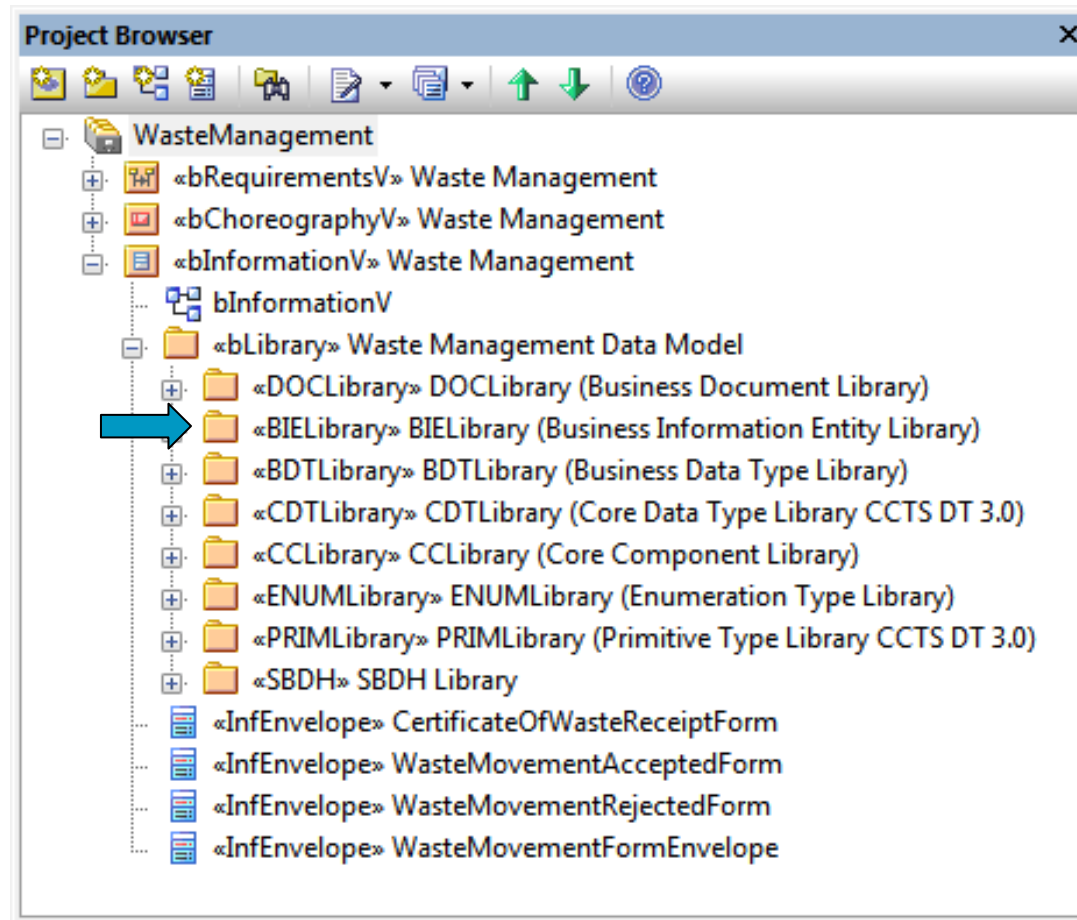
- Business Data Types (BDT) are derived from Core Data Types (CDT) **by restriction**
- **Content Components (CON)** hold the actual information (e.g. 12R17)
- **Supplementary components** provide additional meta information for the content component



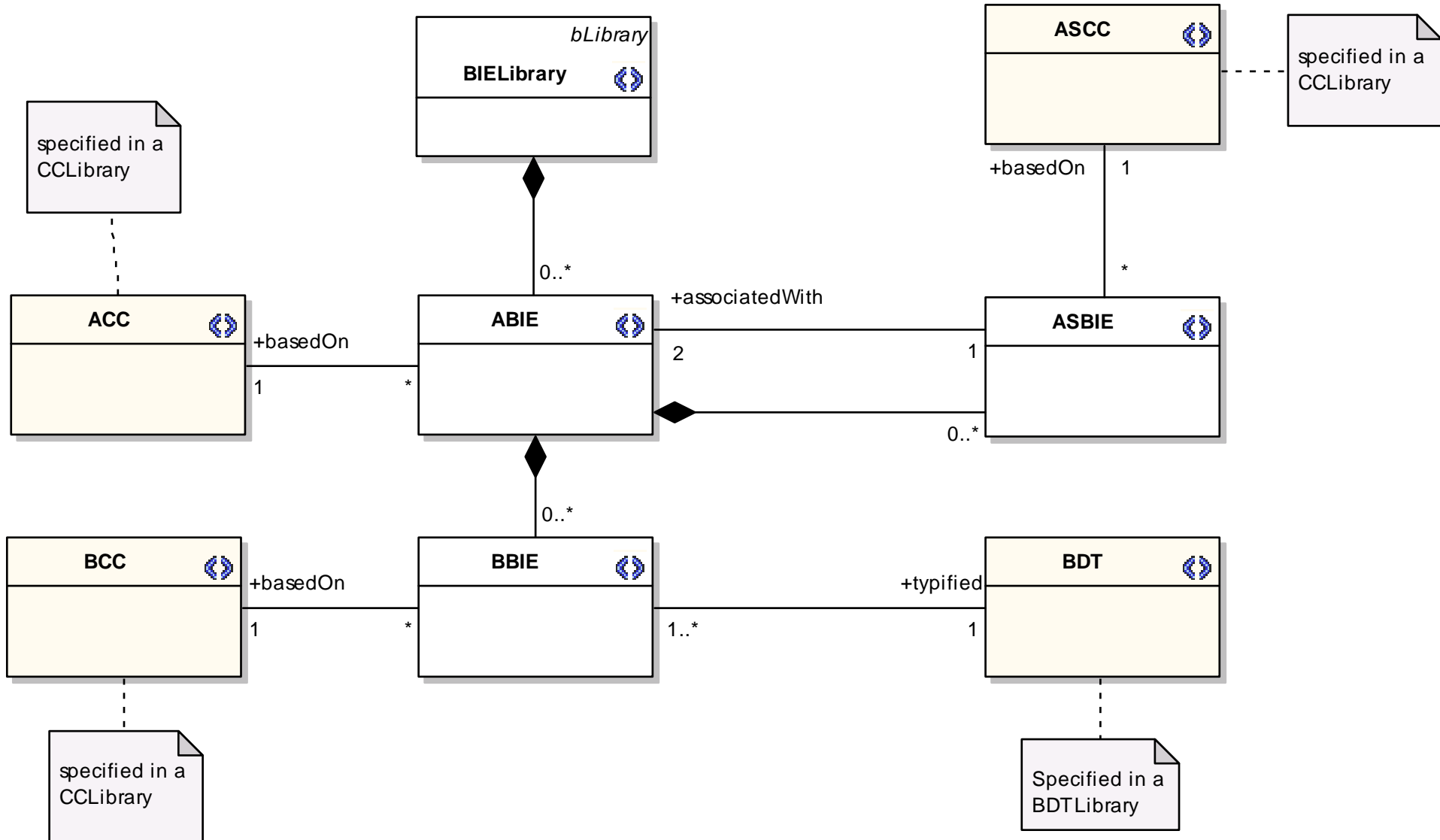
Setting the value domain of content components and supplementary components of Business Data Types (BDT)



BIELibrary – Business Information Entity Library

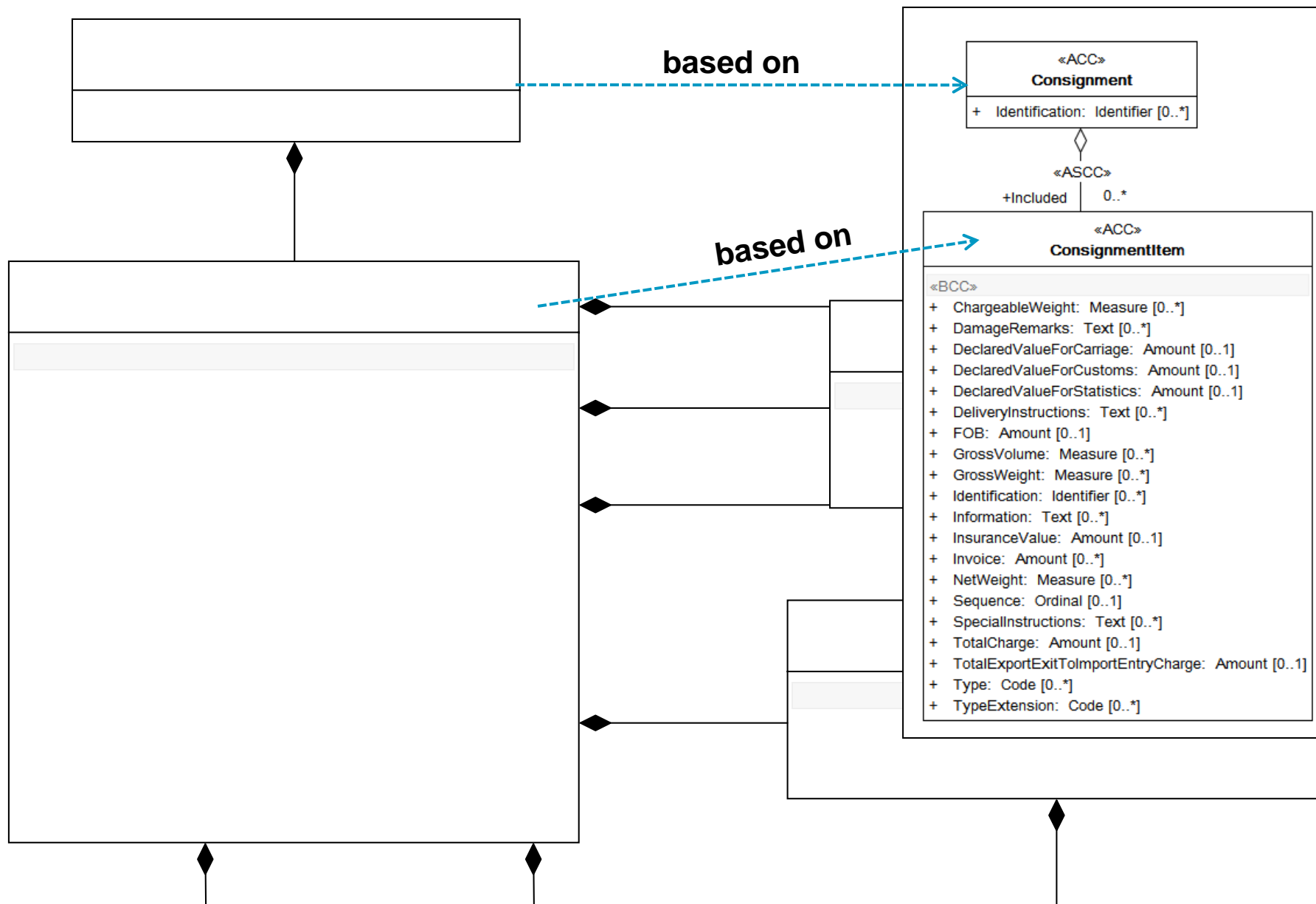


BIELibrary – Business Information Entity Library (conceptual)

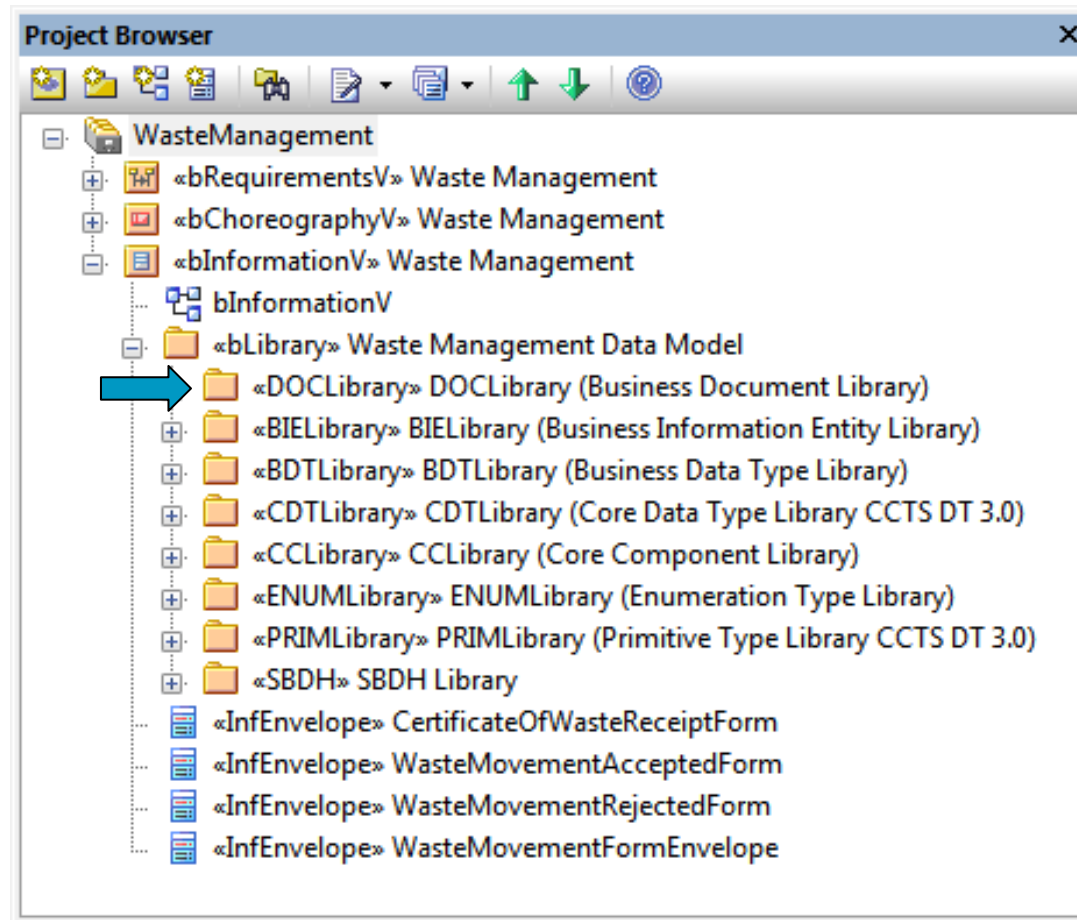


- A business document modeler takes a Core Component from the Core Component Library and tailors it to the specific needs of a specific domain
- The Core Component becomes a **Business Information Entity**
- For each Core Data Type, used to set the value domain of Basic Core Components, a **Business Data Type must be created**
- Business Information Entities are used to assemble business documents
- A BIELibrary may be thought of, a certain repository of reusable business document building blocks
- Since each Business Information Entity is based on a Core Component, **a common understanding of the document semantics is guaranteed**

Business Information Entity Library – **rsa** example



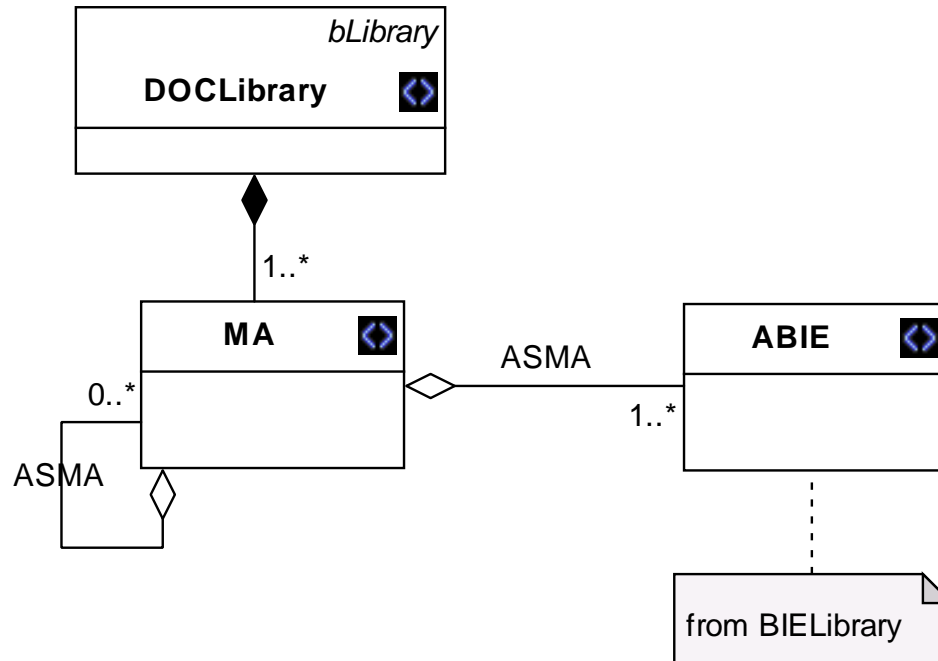
DOCLibrary – Business Document Library



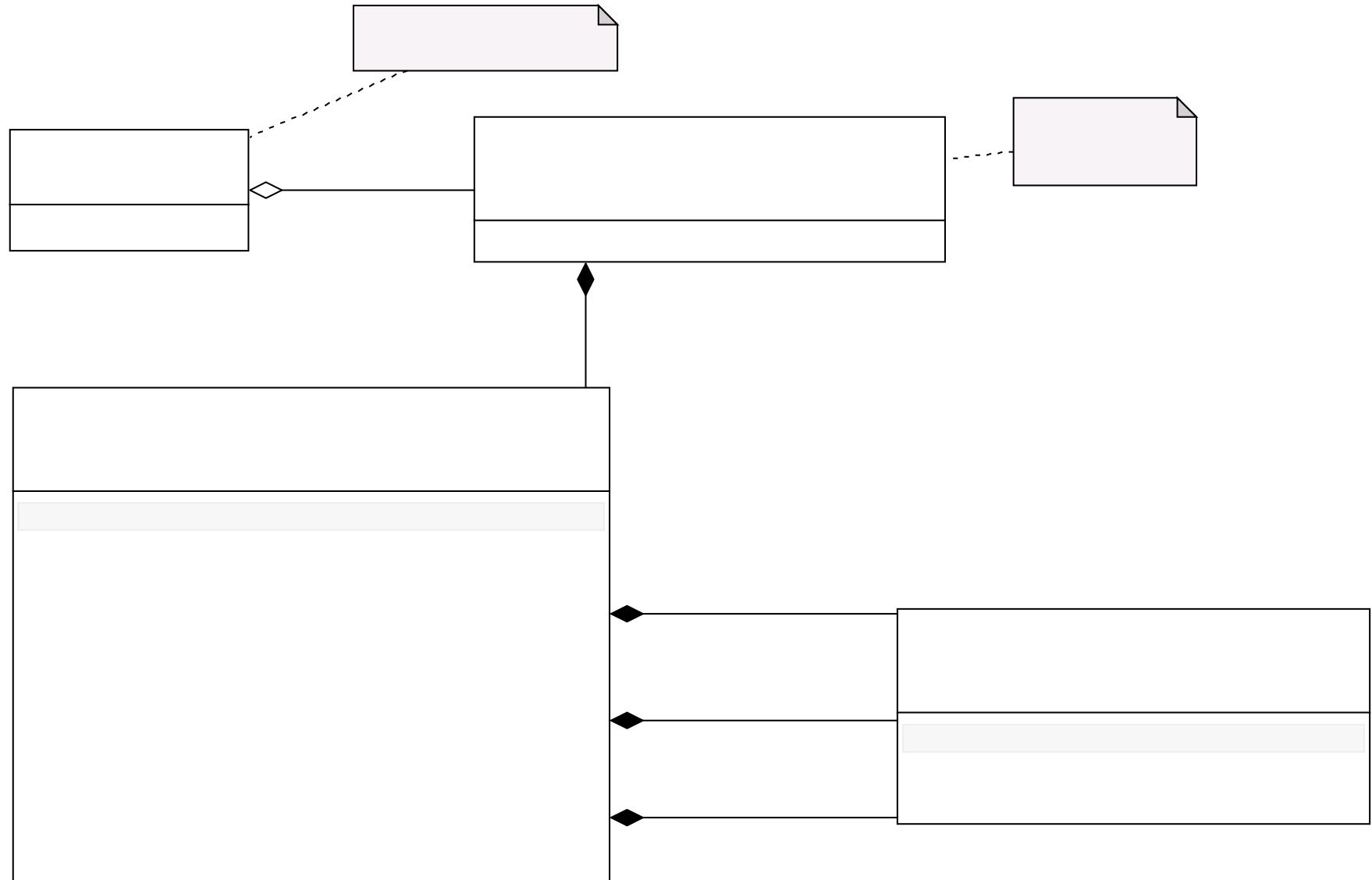
DOCLibrary - Business Document Library – challenges

- CCTS mandates a strict corset
 - Every Business Information Entity concept must be based on the respective Core Component concept e.g. ASBIE must be based on ASCC
 - There must not be a BIE without an underlying CC
- **These restrictions are necessary** to enforce adherence of business document modelers to a common semantic business document base
- Business documents are assembled using Business Information Entities
- Concepts are necessary to **allow an aggregation of different Business Information Entities**
- UPCC introduces two new concepts
 - Message Assembly (MA)
 - Association Message Assembly (ASMA)

DOCLibrary concepts

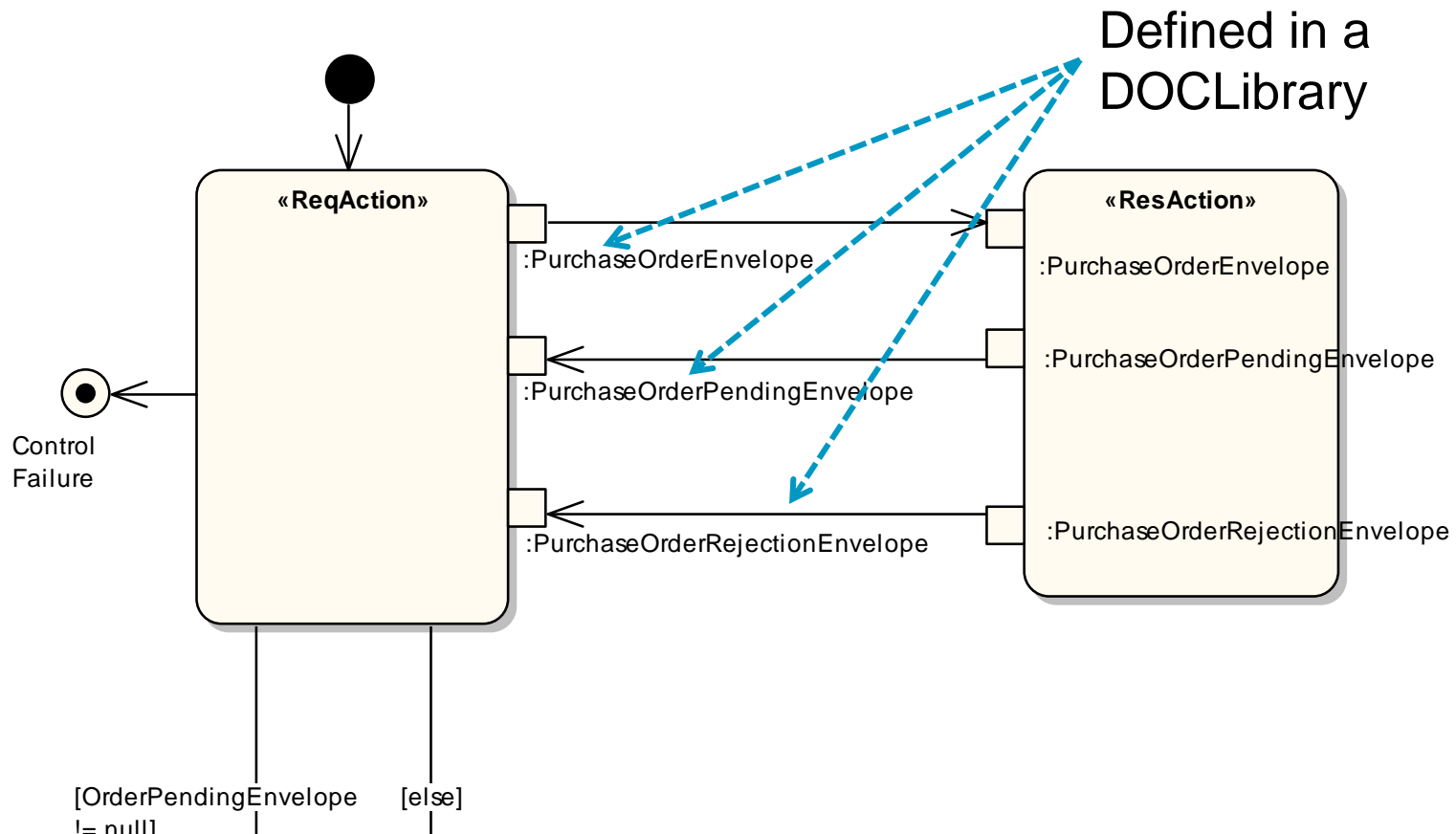


Business Document Library

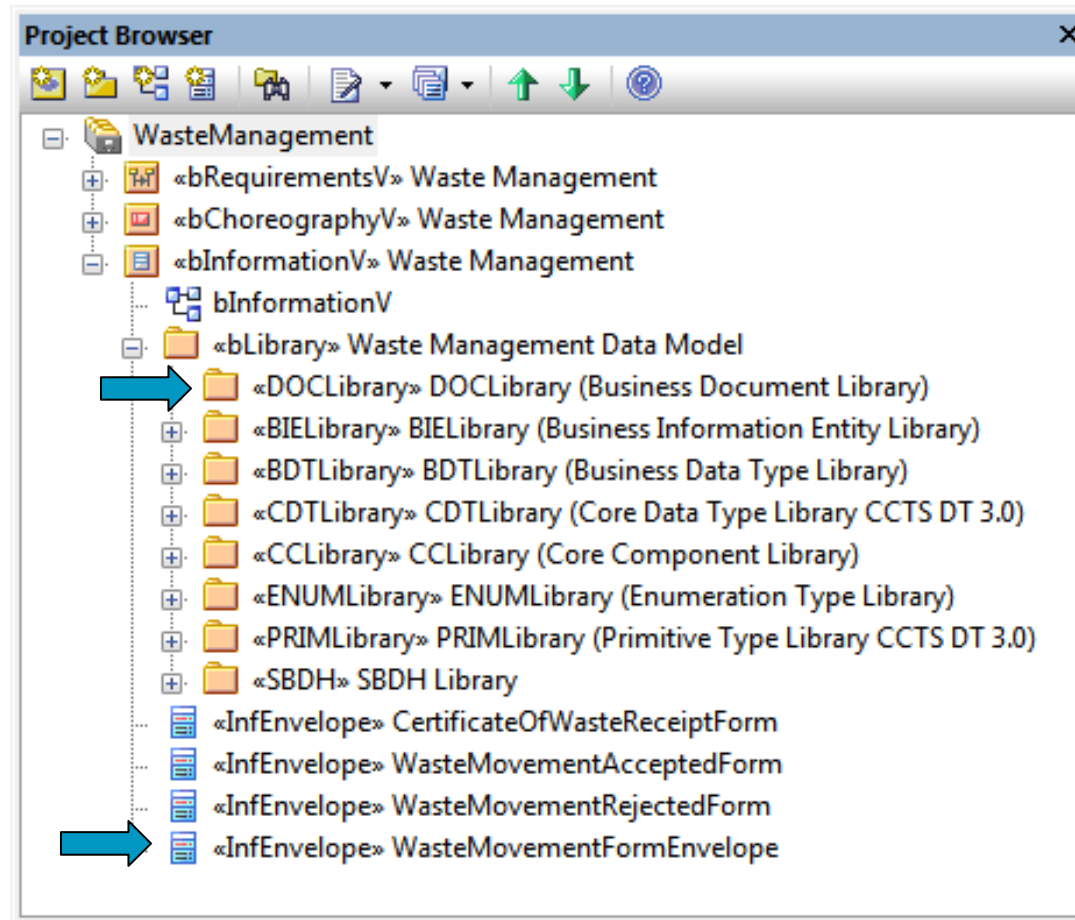


Combining UN/CEFACT's Modeling Methodology (UMM) and the UML Profile for Core Components (UPCC)

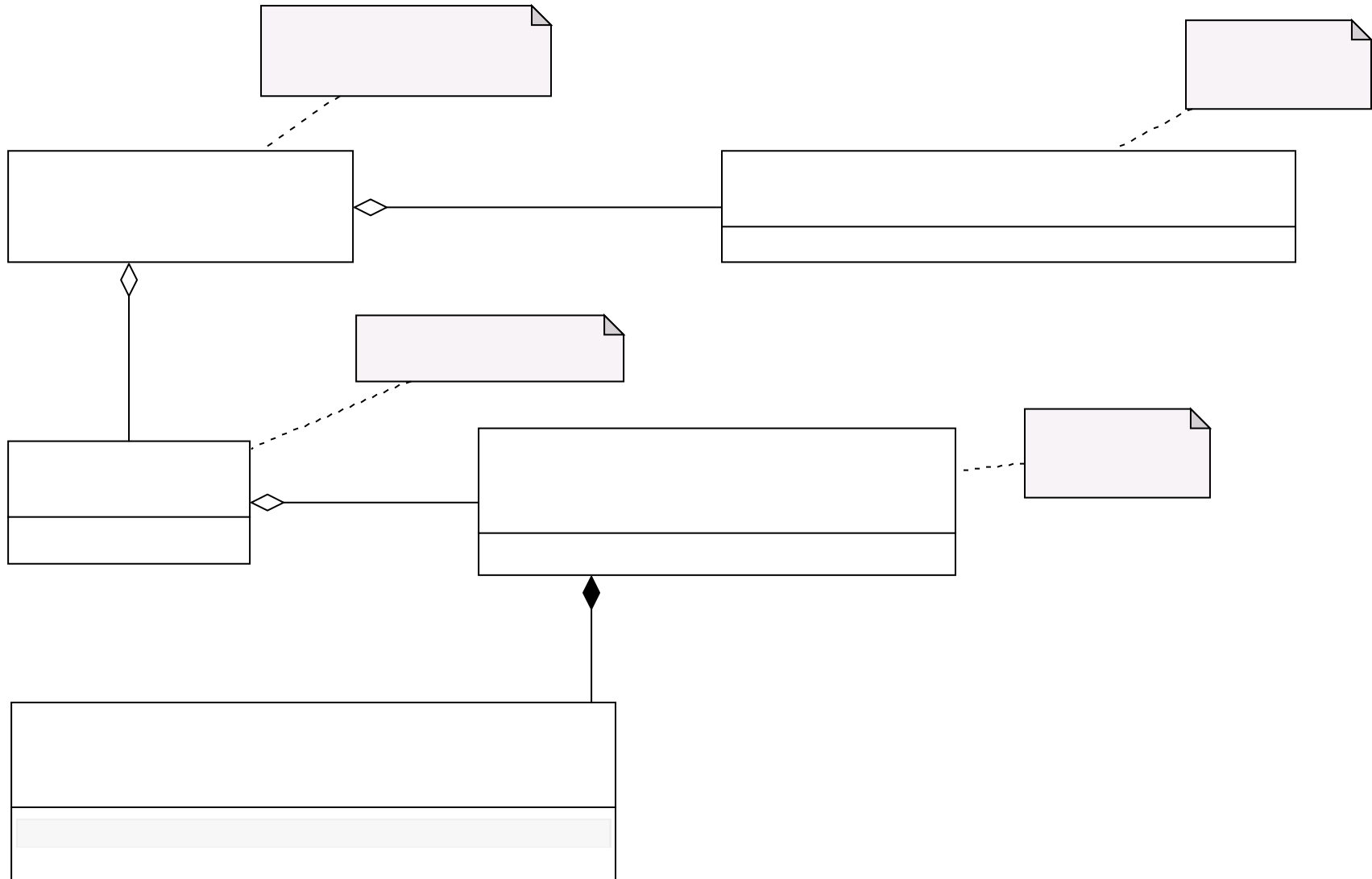
- Business Documents are aggregated in dedicated business document libraries (DOCLibrary)
- Connect the business document artifacts to action pins of UMM business transactions



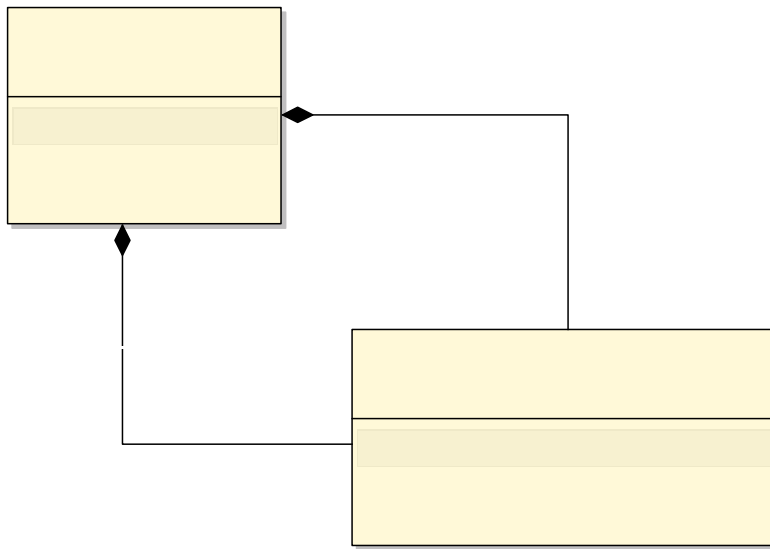
Combining UMM & UPCC



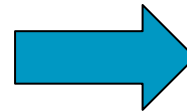
Combining UMM & UPCC cont'd



From conceptual models to deployment artifacts **rsa**



Naming and Design Rules



VIENNA AddIn



```
<xsd:complexType name="US_PersonType">
  <xsd:sequence>
    <xsd:element name="DateofBirth" type="udt1:DateType"/>
    <xsd:element name="FirstName" type="udt1:TextType"/>
    <xsd:element name="US_Work" type="bie1:US_AddressType"/>
    <xsd:element name="US_Private" type="bie1:US_AddressType"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="US_AddressType">
  [...]
</xsd:complexType>
```



VIENNA AddIn

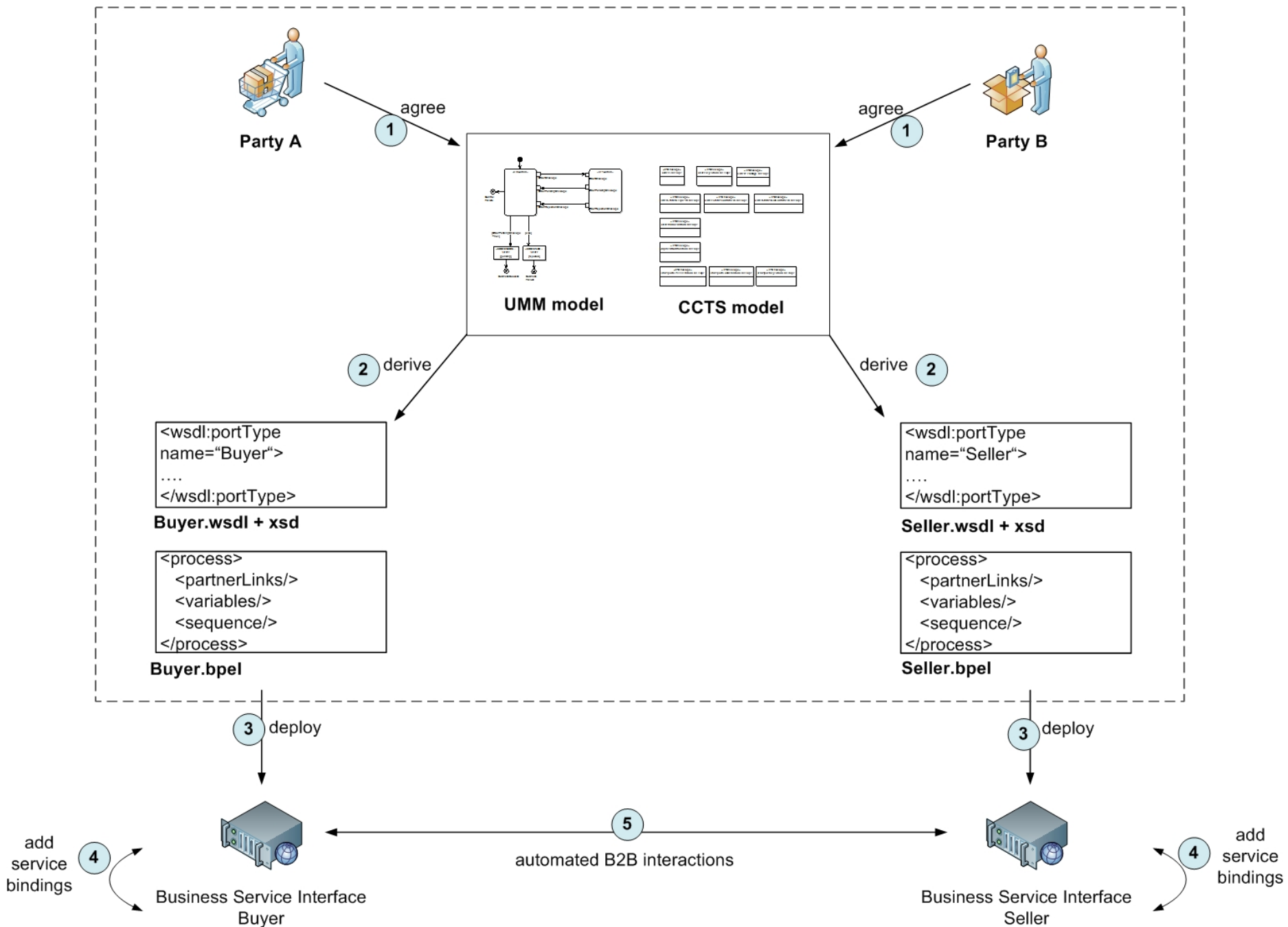
UMM (UN/CEFACT's Modeling Methodology)

- Model validation
- Transformation to choreography languages (BPEL, BPMN)
- Automated model structure generator

UPCC (UML Profile for Core Components)

- Model validation
- Deployment artifact generation (XSD)
- Model-artifact creation wizards
- Core Component Library import

VIENNA Add-In



Thank you for your attention

<Lecturer>

<Name>**Philipp Liegl**</Name>

<Company>**Research Studios Austria**</Company>

<Department>**Research Studio Inter-Organisational Systems**</Department>

<Address>

<Street>**Thurgasse 8/3/20**</Street>

<ZIP>**1090**</ZIP><City>**Vienna**</City>

<Country>**Austria**</Country>

</Address>

<Contact>

<Email>**office.ios@researchstudio.at**</Email>

<Http>**http://www.researchstudio.at**</Http>

</Contact>

<? Presentation status=**“questions”** ?>

</Lecturer>