Feedback from OASIS UBL TC to Draft Core Components Specification 1.8

document id

Version 5.2

May 4, 2002

Editors Bill Burcham, Sterling Commerce
Tim McGrath, UBL LCSC chair
Lisa Seaburg, AEON Consulting

To simplify the reading of our comments, we will refer only to Core Components (CCs). In reality these comments apply to both CCs and Basic Information Entities (BIEs). The issue of context is not covered in our comments.

Specifically, the areas we wish to comment on relate to:

1. **Properties and Their Terms**

2. **Representation Terms, Core Component Types and Basic Core Components**

3. **Consistent Application of Tripartite Naming at the ACC Level and the BCC Level**

4. **The Use of Codes and Identifiers**

1. Properties and Their Terms

There appears to be an imprecise treatment of "properties". While that specification *does* talk extensively about "property terms" – which are part of a "data element name" for a "data element" [NAMING-ISO], we are left to *infer* the existence and makeup of the "property" concept.

We are trying to give "property terms" to things. What things are we trying to give them to? The specification doesn't tell us.

The term "property" is used often in [CCTS][1], but it is never formally defined. Additionally, the term "child field" is sometimes used synonymously to "property", and is also left undefined. Furthermore, neither appears in any of the conceptual diagrams.

Proposal 1

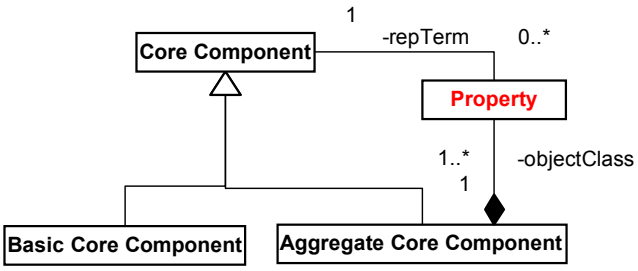
The CC model should explicitly include the concept of *property*.

[1] CCTS Section 5.6 lines 838-851; section 5.6.2 lines 892-914

Property is the model element named by a *property term*. This is similar to the way a *Core Component's* "activity or object"[2] is the model element named by an *object class*.

This concept (property) corresponds to "field" in database models, "attribute" in ER modeling, "member" in Java, "child element" in XML, and "attribute" in UML.

Figure 6-1 in [CCTS] suggests the need for this concept. This proposal simply involves elaborating the aggregation relationship from ACC to itself labeled "Contains", with an interposing class (called Property). An Aggregate Core Component has many properties. Each property has a name. A property has a type (drawn from the list of Core Components). Here's how[3] the Property concept fits into the metamodel:



Properties are the means by which we model one-way relationships between types. To capture the sense of this unidirectional association we must assign role names to the types related by a property. Adopting the [NAMING-ISO] terminology, the ACC that contains the property is referred to as the Object Class for that property. The CC that defines the type of the property is referred to as the Representation Term for that property. This Representation Term describes the form of the set of valid values of a data element [NAMING-ISO].

For example, imagine an ACC called "Address". This ACC might have properties: "Street" of type "Text", and "Country" of type "Code". Summarizing, these properties could be defined thus:

Object Class	***Property Name/Term***	*Representation Term*
Address	Street	Text
Address	Country	Code

This proposal formalizes the prose already in the specification.

[2] CCTS lines 2162-2163

[3] It's worth noting that the proposed design as exemplified in the diagram is an instance of the "Composite Pattern" from [GOF].

"Property Term - This identifies one of the characteristics belonging to the Object Class."[4], and *"… represents the distinguishing characteristic or property…"*[5]

Proposal 2

A property's name (i.e. Property Term) should reflect the role played by that property's content *relative to* the Object Class/Aggregate Core Component in which that property is declared.

This new, formal concept of property allows us to identify (name) a Core Component (either a Basic or subsidiary Aggregate Core Component) within the Object Class/Aggregate Core Component that contains it.

For example, the object class 'Shipping' might have a 'From' property and a 'To' property. Each of these properties represents a 'Location'.

The terms ('From' and 'To') obviously reflect the role played by the respective 'Locations'. Without the concept of property, there is no way to distinguish these two *uses* of the object class Location within the (Aggregate) object class 'Shipping', as in…

Object Class	*Property Name/Term*	*Representation Term*
Shipping	From	Location
Shipping	To	Location

Proposal 3

The [NAMING-ISO] term "Data Element" is identical to the "Property" concept described in Proposal 1 and Proposal 2

The [NAMING-ISO] standard governs specification and standardization of data elements. The definition from that standard:

data element: A unit of data for which the identification, meaning, representation and permissible values are specified by means of a set of attributes (ISO/IEC 11179-3)

Property as described in Proposal 1 and Proposal 2 is exactly such a data element.

Proposal 4

[4] CCTS lines 2167-2168

[5] CCTS lines 1115-1116

A (tripartite) Data Element Name [NAMING-ISO] for a Property is constructed from the Property's ObjectClass, PropertyName/Term and RepresentationTerm

The mapping of [NAMING-ISO] to the proposed Core Components model is trivial:

[NAMING-ISO] Data Element Name Components	Proposed CC Model
Object Class Term	The ACC playing the "Object Class" role relative to the Property
Property Term	The name of the Property
Representation Term	The CC playing the "Representation Term" role relative to the Property.

2. Representation Terms, Core Component Types and Basic Core Components

In resolving Representation Terms (table 6-1 in [CCTS]) with the Core Component Types (table 8-1 in [CCTS]) in the production of the UBL vocabulary we note that certain concepts are identical:

- Amount and Amount.Type
- Graphic and Graphic.Type
- Indicator and Indicator.Type
- Picture and Picture.Type

Others are unaccountably different in wording but mean the same thing:

- Measure and Measure.Type

Others are essentially identical except for mention of supplementary components:

- Code and Code.Type
- DateTime and DateTime.Type
- Identifier and Identifier.Type (but Identifier carries a warning about when to use Name instead)
- Quantity and Quantity.Type

And many are related by generalization/specialization:

- Date isa DateTime.Type
- Name isa Text.Type
- Percent isa Numeric.Type
- Rate isa Numeric.Type

(Definitions of these concepts, gleaned from [CCTS] and presented side-by-side for comparison may be found in *Appendix C: Comparing Representation Term and Core Component Type Definitions*.)

There is clearly significant overlap between the two sets of concepts. Given this significant overlap we view the use of two classifications as suspect.

As the UBL team started trying to use the two sets of classifications, we have been realizing that there are some significant failings in this system.

One example of this is a Location Code, a common piece of data that cannot be described by a set of enumerated values - what we traditionally and typically think of as a 'code list' - even though it's business function is that of a code. [CCTS] does not account for this phenomenon. Because it functions as a "code", it has a semantic primitive RT of "Code" - this means that it has a "Code" CCT, which allows you to point to a code-list and supporting properties.

In addition, we would see value in being able to model a number of properties, each with a Representation Term “Identifier”, where some properties might have Core Component Type “TextType” and others “NumericType”. Unfortunately, as specified, Identifiers have to be of IdentifierType, so even that example is impossible.

As a final example of the need for more accurate semantic primitive definitions, we can use the degree of precision of a price. This is a fundamental kind of data-type issue, since the degree of precision in prices defines the tolerances used in the calculations for essential business processes such as Order/ASN/Invoice reconciliation (aka "book-keeping"). If we are to describe a "price" using the current system, here is what we would know about it:

CCT = "AmountType" (which gives us a number and a currency code)

The degree of precision of the price cannot be specified in the semantic model, given these capabilities. All monetary amounts are the same. But in reality, prices have a very different specificity than some other monetary amounts. This means we cannot simply assume a single precision for all monetary amounts. Precision is not syntax-specific. It is a critical property of the business data itself. In other words, there is a need to capture in the model some distinction between a price and another kind of monetary amount, since they have different requirements in terms of how they are represented.

Even if the multiplicity between Representation Term and CCT (see figure 6-1 in [CCTS]) were changed appropriately, we don’t see sufficient value in the distinction to merit its cost.

Proposal 5

Merge the list of Representation Terms and Core Component Types.

If unification were undertaken, the generalization/specialization cases would need to be considered since right now there's no notion of "inheritance" in the metamodel. See *Appendix B: Sample Unified Representation Term / Core Component Type Hierarchy* for a sample hierarchy.

Proposal 6

Consider that merged list of Representation Terms and Core Component Types "Basic Core Components".

The relationship from BCC to Representation Term labeled "Is Based On" and the relationship from Representation Term to CCT labeled "Is Derived From" are both arguably representing subtyping (BCC is a subtype of RT is a subtype of CCT) per figure 6-1 in [CCTS].

Subtyping is a valuable concept. Unfortunately, applying it in such a limited way provides little benefit at some cost in clarity. We therefore recommend that since RT and CCT lists should be merged (Proposal 5) and since there is little value in a BCC's being declared a subtype of a CCT (when for instance an ACC is prohibited from being declared a subtype of some CC), that all three concepts (CCT, RT and BCC) be collapsed into a single concept: Basic Core Component.

Proposal 7

A Basic Core Component will consist of a Primary Component and Supplementary Components

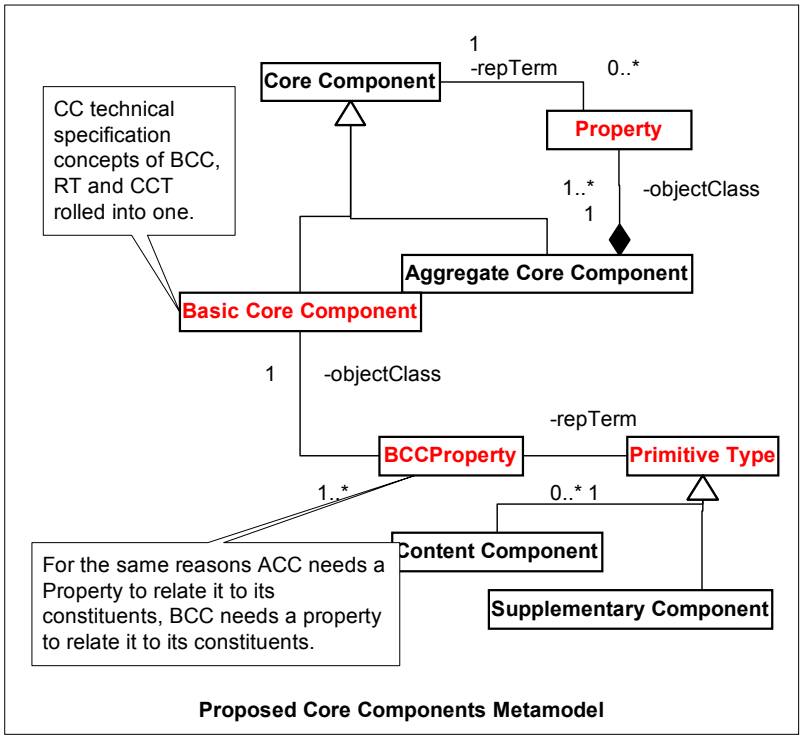
This new notion of Basic Core Component will carry the old structure of Core Component Type, i.e. it will have a Primary Component and Supplementary Components.

Proposal 8

A Basic Core Component will relate to its Primary and Supplementary Components through a BCCProperty.

For all the reasons why an ACC needs a Property to relate to its constituents, so also, a BCC needs a property (w will call BCCProperty) to relate to its constituents. See Proposal 1 and Proposal 2 for our justification of the idea of Property.

We are now in a position to present a complete formulation of the proposed Core Components Metamodel – a revision to Figure 6-1 in [CCTS]):



Proposed Core Components Metamodel

3. Consistent Application of Tripartite Naming at the ACC Level and the BCC Level

One problem solved by the proposed metamodel changes is that it is now possible to consistently and clearly apply [NAMING-ISO] tripartite naming to properties of "leaf-level" components (Basic Core Components) just as easily as we do to properties of "interior" components (Aggregate Core Components). Indeed, there is a simple elegance in the recursive nature of this that makes it attractive.

For example the ACC called "Department" has a property called "Manager" of type "Employee". Employee is an "interior" component, it is itself an ACC and might have another property, called "Mentor", also of type "Employee". Here are the property definitions for the Employee example. These directly suggest the tripartite names:

Object Class	*Property Name/Term*	*Representation Term*
Department	Manager	Employee
Employee	Mentor	Employee

Imagine further that an Employee has a property called "StartDate" of type "Date". Let's assume that Date is a Basic Core Component as defined by Proposal 6, Proposal 7 and Proposal 8. That property of Employee is described thus:

Object Class	*Property Name/Term*	*Representation Term*

Employee	StartDate	Date

But isn't it also important to describe the elements of Date (a Basic Core Component)? Let us say that a Date has a Content Component of some Text type and that it has a Supplementary Component of come code type identifying a calendrical scheme[6]. We could describe those elements thus:

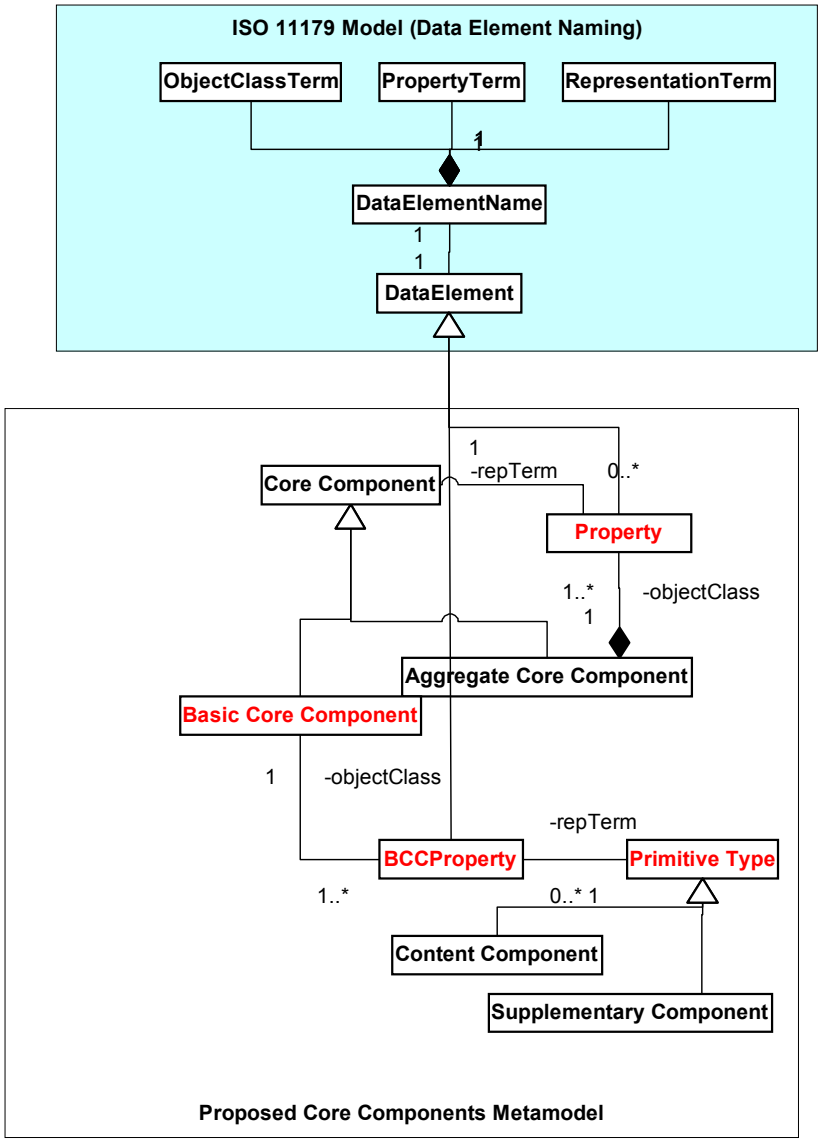
Object Class	*Property Name/Term*	*Representation Term*
Date	ContentComponent	Text
Date	Scheme	CalendricalSchemeCode

The regular structure of the proposed metamodel coupled with the elimination of the concept "Representation Term" from it, allows us to apply tripartite naming consistently.

We start to see that, if we remove the differentiation between Basic Core Components and types, we can simplify the application of the tripartite naming scheme and remove the need for artificial sub-classification schemes.

This new mapping of [NAMING-ISO] to [CCTS] can now be shown graphically:

[6] This is an artificial example for the purpose of illustration. There would probably be more appropriate ways to represent the supplementary components of dates.



You can see that the two kinds of property (Property and BCCProperty) are properly viewed as "kinds of" Data Element. Also note the role names on the associations with properties are suggestive of the Data Element Name components. Both Property and BCCProperty have associates in the role of "repTerm" (short for Representation Term) and "objectClass". There are some other outcomes of the proposal worth noting here:

The proposed model eliminates the "Details" Representation Term.

Since an element of an ACC may be of any CC type, and the Representation Term in that case is the name of that CC type we no longer need the distinguished (and empty) Representation Term, "Details". Instead, as shown in the Employee example in section 3 *Consistent Application of Tripartite Naming at the ACC Level and the BCC Level* under the proposed model we now see a meaningful CC type name in place of "Details".

The set of Representation Terms includes ACC's under the proposed model.

It is important too, to note that under this proposal the set of all Representation terms appearing in the data dictionary will include both “leaf-level” components (BCC’s) and “interior” components (ACC’s). This is in contrast to [CCTS]. This list of all Representation Terms is still “controlled” – the key difference is that under this proposal the list will include ACC’s.

4. The Use of Codes and Identifiers

No one issue has caused as many problems as the application of these two concepts. Unlike other proposals, this is not a meta-model issue; it is an issue of content and terminology.

There appears to be much confusion about the terms ‘codes’ and ‘identifiers’. The issue of when to declare a Basic Core Component either a Code or an Identifier still needs clarification. Getting the etymological roots of the terms correct seems a reasonable first step. As has been evidenced in the various debates on these issues, when carried down to enumeration and validation, it gets more complicated. For example, there is a natural tendency to want to enumerate for validation purposes small sets of things that we have been considering "codes", which are actually "identifiers".

Therefore, at this stage we would encourage the CCTS to leave enumeration and validation out of the picture and concentrate on getting the semantics correct.

Proposal 9

There is a need to clarify the semantics of the terms ‘code’ and ‘identifier’.

We would suggest the following definitions:

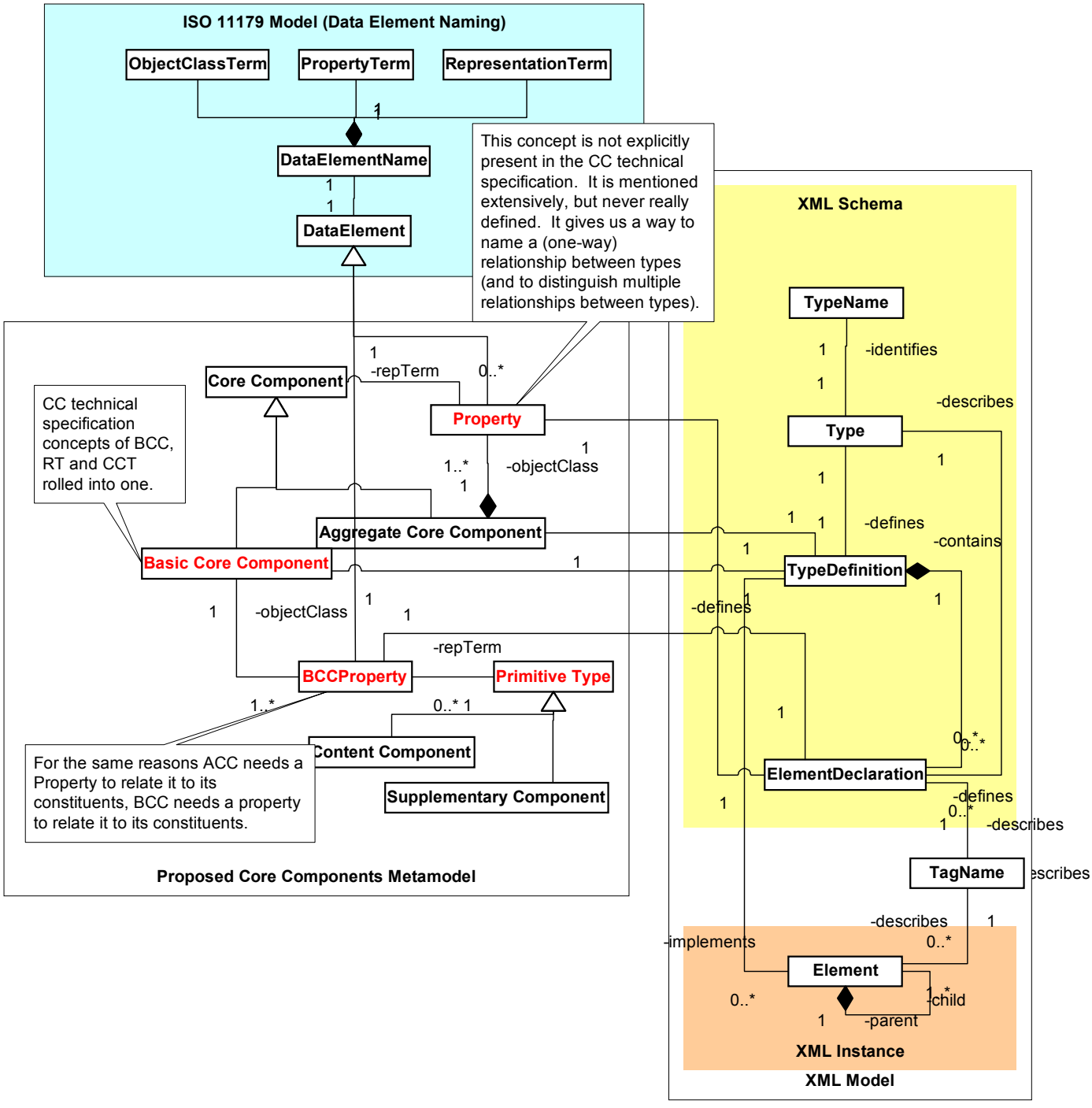
Code: a system of words, figures or symbols used to (exactly) represent others.
(This definition comes direct from the Oxford English Dictionary. We have omitted the following phrase 'especially for the purposes of secrecy' which came after the word 'others'. The word 'exactly' is an addition.)

Identifier: that which establishes the identity of (something).
(This definition is derived from the definition of 'identity' in the Oxford English Dictionary (OED).)

Appendix A: Proposed Core Components Metamodel

In the body of this document we developed a revised Core Components Metamodel. Much of the discussion hinged on the relationship of that model to that of [NAMING-ISO]. As we went along, we developed increasingly detailed pictures of those models and their relationships.

The final metamodel, a reformulation of Figure 6.1 *Core Components Metamodel* [CCTS] as shown in section *3 Consistent Application of Tripartite Naming at the ACC Level and the BCC Level*. In addition, we've shown the mapping of that model onto XML [XML] and XSD [XSD]. We believe wholeheartedly in the ethic of a metamodel that is independent of syntax binding. On the other hand we feel it is incumbent upon us to verify that *some* binding is possible. Hence the mapping to XML.


ISO 11179 Model (Data Element Naming)
ObjectClassTerm
PropertyTerm
RepresentationTerm
DataElementName
DataElement
This concept is not explicitly present in the CC technical specification. It is mentioned extensively, but never really defined. It gives us a way to name a (one-way) relationship between types (and to distinguish multiple relationships between types).
XML Schema
TypeName
-identifies
-describes
Type
-defines
-contains
TypeDefinition
ElementDeclaration
-describes
TagName
-describes
-implements
Element
-child
-parent
XML Instance
XML Model
Core Component
-repTerm
Property
-objectClass
CC technical specification concepts of BCC, RT and CCT rolled into one.
Aggregate Core Component
Basic Core Component
-objectClass
-repTerm
BCCProperty
Primitive Type
Content Component
Supplementary Component
For the same reasons ACC needs a Property to relate it to its constituents, BCC needs a property to relate it to its constituents.
Proposed Core Components Metamodel

Appendix B: Sample Unified Representation Term / Core Component Type Hierarchy

Here is an example (not a proposal) for such a unified hierarchy:

```
NumericType
   AmountType
   MeasureType
     QuantityType
   PercentType
   RateType
GraphicType
   PictureType
IndicatorType
CodeType
DateType
   DateTimeType
IdentifierType
TextType
   NameType
```

Appendix C: Comparing Representation Term and Core Component Type Definitions

This section presents Representation Term and Core Component Type definitions gleaned from [CCTS]. The definitions are grouped for easy identification of their similarities.

Amount definitions:

RT: A number of monetary units specified in a currency where the unit of currency is explicit or implied.

CCT: A number of monetary units specified in a currency where the unit of currency is explicit or implied.

Code definitions:

RT: A character string (letters, figures or symbols) that for brevity and / or language independence may be used to represent or replace a definitive value or text of an attribute. Codes usually are maintained in code lists per attribute type (e.g. color).

CCT: A character string (letters, figures or symbols) that for brevity and/or language independence may be used to represent or replace a definitive value or text of an attribute together with relevant supplementary information.

Date definitions:

RT (Date): A day within a particular calendar year (ISO 8601).

RT (DateTime): A particular point in the progression of time (ISO 8601).

CCT (DateTime): A particular point in the progression of time together with relevant supplementary information.

Graphic definitions:

RT: A diagram, graph, mathematical curves, or similar representation.

CCT: A diagram, graph, mathematical curves, or similar representation.

Identifier definitions:

RT: A character string used to establish the identity of, and distinguish uniquely, one instance of an object within an identification scheme from all other objects within the same scheme. [Note: Type shall not be used when a person or an object is identified by its name. In this case the Representation Term ?Name? shall be used.]

CCT: A character string to identify and distinguish uniquely, one instance of an object in an identification scheme from all other objects within the same scheme together with relevant supplementary information.

Indicator definitions:

RT: A list of two, and only two, values that indicate a condition such as on/off; true/false etc. (synonym: ?Boolean?).

CCT: A list of two, and only two, values, which indicate a condition such as on/off; true/false etc. (synonym: ?Boolean?).

Measure definitions:

RT: A numeric value determined by measuring an object. Measures are specified with a unit of measure. The applicable unit of measure is taken from UN/ECE Rec. 20.

CCT: The size, volume, mass, amount or scope derived by performing a physical measure together with relevant supplementary information.

Name definitions:

RT: A word or phrase that constitutes the distinctive designation of a person, place, thing or concept.

CCT: (Name RT is of Text.Type, which is defined as A character string with or without a specified language.)

Percent definitions:

RT: A rate expressed in hundredths between two values that have the same unit of measure.

CCT: (Percent RT is of Numeric.Type, which is defined as A representation of a number.)

Picture definitions:

RT: A visual representation of a person, object, or scene.

CCT: A visual representation of a person, object, or scene.

Quantity definitions:

RT: A number of non-monetary units. It is associated with the indication of objects. Quantities need to be specified with a unit of quantity.

CCT: A number of non-monetary units together with relevant supplementary information.

Rate definitions:

RT: A quantity or amount measured with respect to another measured quantity or amount, or a fixed or appropriate charge, cost or value e.g. US Dollars per hour, US Dollars per EURO, kilometer per liter, etc. (Taken from V1.6; PDF for V1.8 is broken.)

CCT: (Rate RT is of Numeric.Type, which is defined as A representation of a number.)

References

CCTS	*UN/CEFACT Draft Core Components Specification, Part 1*, 8 February, 2002, version 1.8
GOF	*Design Patterns,* Gamma, et al. ISBN 0201633612
NAMING-ISO	*ISO/IEC 11179,* Final committee draft, Parts 1-6.
XML	*Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation, October 6, 2000
XSD	*XML Schema Part 0: Primer*, W3C Recommendation, 2 May 2001