UBL Feedback from OASIS UBL TC to Draft Core Components Specification 1.8

document id
Version 4.1 editors Bill Burcham, Sterling Commerce
Tim McGrath, UBL LCSC chair
Lisa Seaburg, AEON Consulting
May 1st, 2002

The UBL group believe that, whilst the current CCTS provides a strong basis for good semantic modeling and definition of Core Components and Basic Information Entities, some modifications and clarifications would make it even better. Our comments are heavily based on our initial experience at applying the CCTS to the development of the UBL library of BIEs.

Some of these modifications may appear significant, but we feel it necessary to raise these matters sooner rather than later, whilst the implementations of ebXML libraries (such as UBL) are still under development.

Having said that, we are mindful of the need for the CCTS work to mature and move forward. We do not want to detract from the team's momentum and hope that you will consider many of these proposals as simplifications rather than complications to your work.

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. Representation Terms and Core Component Types
2. The use of Representation Terms for Aggregate Core Components
3. Properties and their terms
4. The use of Codes and Identifiers

1. Representation Terms and Core Component Types

It is becoming clear that the differentiation between Representation Term and Core Component Type is confusing and does not add any semantic meaning to definitions.

Proposal 1

> **We suggest that the definition of Representation Term be clearly aligned with the ISO11179-5 definition.**

This [NAMING-ISO] makes three key points about representation terms:
1. A representation term is a component of a data element name which describes the form of representation of the data element.
2. Each term is developed from a controlled word list or a taxonomy.
3. This term describes the form of the set of valid values of a data element.

Proposal 2

The controlled word list or taxonomy referred to in ISO 11179 should function as "semantic primitives".

We understand this may have been the original intention of the CC group in ebXML. (The list may need to be altered slightly to include some missing types, but will not undergo wholesale expansion). This indicates what the business purpose of the data is, in an abstract sense, wholly separate from how it will be represented when syntax bound.

Proposal 3

That this set of Representation Terms (proposal 2) removes the need for Core Component Types.

We no longer need Core Component Type in the CC metamodel.

<bill> Why don't we keep Core Component Type in the model and make it stand for the "semantic primitives" (rather than throwing out CCT and keeping RT). Let's keep Representation Term as part of the dictionary entry name.</bill>

2. The use of Representation Terms for Aggregate Core Components

The current Core Component Types (or Representation terms if we adopt the proposal above) are aggregate data structures. These structures are re-used in many different Basic Core Components (as codes, amounts, etc.).

It would seem that we could view the use of Aggregate Core Components in the same manner as these Representation Terms. When we define an Aggregate Core Component we are actually establishing new aggregate data structures suitable for re-use (e.g. Address, Period, Contact, etc). This is a higher-level version of a Representation Term. Therefore, perhaps these Aggregate Core Components should be treated as higher-level aggregations of the Basic Core Component Representation Terms.

For example, we may have a Basic Core Component called “Start. Date Time” which has as “Date Time” Representation Term. This means the CC is represented as an aggregation of “Date Time. Content” and “Date Time. Format. Text”. This Basic Core

Component can be used within an Aggregate Core Component such as “Period” to become “Period. Start. Date Time” together with other Basic Core Components, such as “Period. End. Date Time”. This Aggregate Core Component (“Period”) is rather like a high-level Representation Term, except that it describes the form of the set of valid values of the *aggregation* rather than the *individual data element*. So when we have a specific instance of our Aggregate Core Components we should be able to use this representation. That is, “Contract. Validity. Period” should have a Representation Term of “Period”, i.e. “a Period represents the Validity of the Contract”. This conforms with the idea that “a [Representation Term] represents the [Property Qualifier, Property Term] of the Object Class.”

Proposal 4

> **Allow Aggregate Core Components to be re-used as Representation Terms**.
>
> Whenever these Aggregate Core Components are re-used, they would have a Representation Term that reflected the form of the set of valid values rather than the meaningless ‘Details’ currently required. This would simplify the defining and enrich the naming of Aggregate Core Components, especially where they are used within other Aggregate Core Components.
> These ‘aggregate’ Representation Terms still conform to a controlled list – the list of Aggregate Core Components. <eve> are we really advocating that there be a closed list to which only a single organization can add?</eve>

3. 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 “dictionary entry 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 appear in any of the conceptual diagrams.

Proposal 5

> **The CC model should include the concept of *property***.
>
> Property is the model element named by a *property term*. This is similar to the same way a *Core Component’s “*activity or object”[2] is the model element named by an *object class.*

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

[2] CCTS lines 2162-2163

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

Proposal 6

Specify that the name of a property (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.

Knowing its 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. That is, its name reflects the property's relationship to the other elements of the CC meta-model.

For example, the property of 'address' can belong to the object class, Party as can the property of 'name'. The terms (Address and Name) obviously reflect the role played by that property's content. Party cannot have more than one Name-proprty or Address-property , unless the property term is qualified (e.g Party Billing Address, Party Shipping Address).

Both these property can also belong to the object class, Location. So their names must be relative to the Object Class in which they are declared..

This proposition formalizes the prose already in the specification.

"Property Term - This identifies one of the characteristics belonging to the Object Class."[3], and *"… represents the distinguishing characteristic or property…"*[4]

Proposal 7

A "Representation Term Property" is needed to relate a Representation Term to the (Content and Supplementary) Components it contains.

For the same reasons a property is needed to relate an ACC to the BCC's it contains, a property is needed to relate a Representation Term (as defined by Proposal 2) to the components it contains.

As with Proposal 6, a Representation Term Property's name should reflect the role played by that property's content *relative to* the Representation Term in which that property is declared.

<eve> is it really necessary to give a different name to RT properties, instead of just explaining how they should be considered (regular) properties? This whole framework is starting to look more and more recursive/elegant; why not encourage it in that direction even further?</eve>

[3] CCTS lines 2167-2168

[4] CCTS lines 1115-1116

Proposal 8

Figure 6.1, the Core Components Metamodel diagram (and its deriviatives), should be updated to reflect these changes.

A revised diagram is given as Appendix A. to this document.

4. The Use of Codes and Identifiers

No one issue has caused as many problems as the application of these two concepts.

Whilst being simplified by accepting the proposal to eliminate Core Component Types (Proposal 3), the issue of when to declare a Basic Core Component a Code or an Identifier still needs clarification. 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'. Getting the entomological 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'.

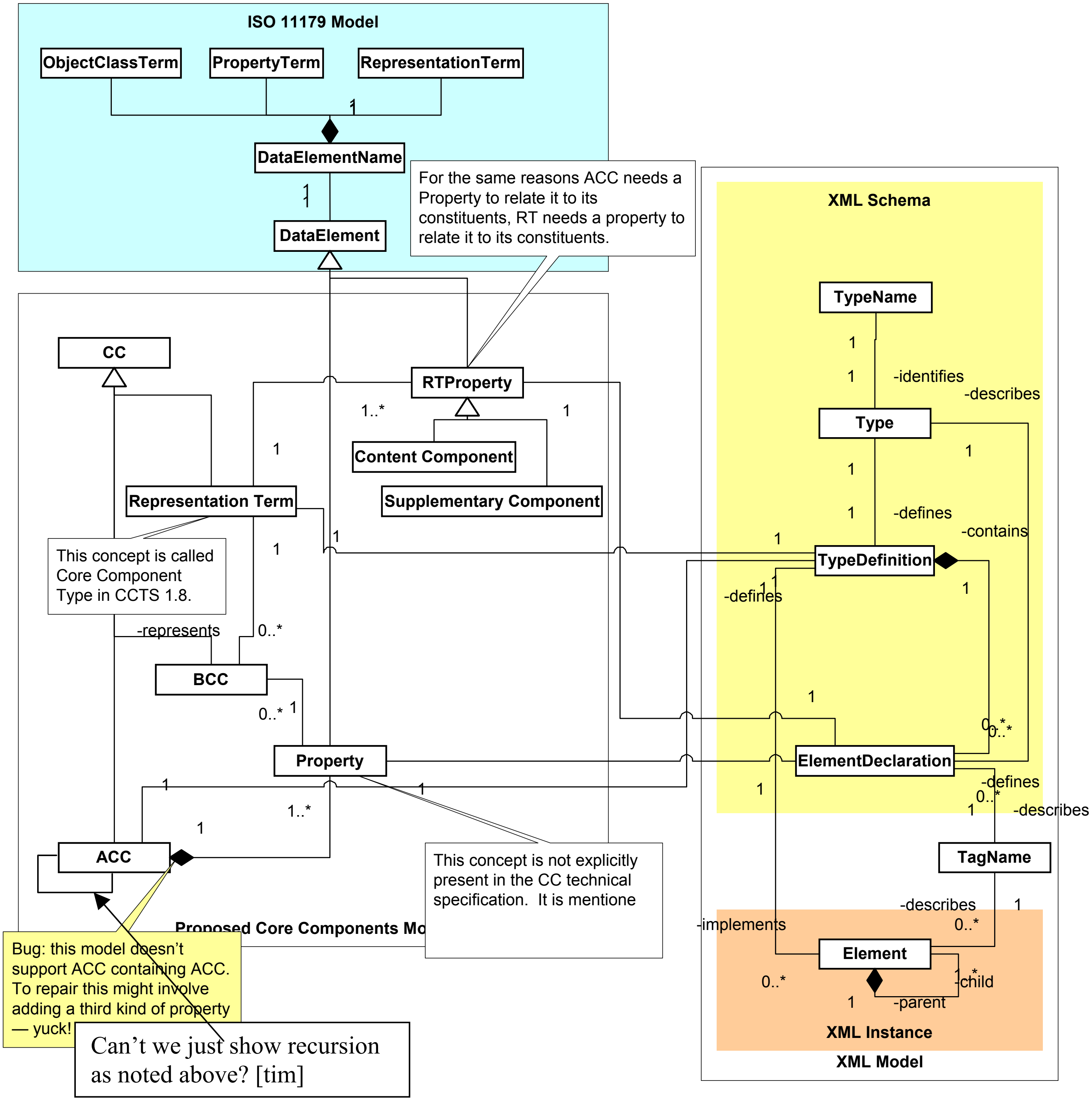
We would suggest the following defintions:

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

As a result the proposals, Figure 6.1 *Core Components Metamodel* should now be as shown in the "Core Components Metamodel" box in this diagram:



For the same reasons ACC needs a Property to relate it to its constituents, RT needs a property to relate it to its constituents.

This concept is called Core Component Type in CCTS 1.8.

This concept is not explicitly present in the CC technical specification. It is mentione

Bug: this model doesn't support ACC containing ACC. To repair this might involve adding a third kind of property — yuck!

Can't we just show recursion as noted above? [tim]

References

CCTS	*UN/CEFACT Draft Core Components Specification, Part 1*, 8 February, 2002, version 1.8	
NAMING-ISO	*ISO/IEC 11179,* Final committee draft, Parts 1-6.	