Feedback from OASIS UBL TC to Draft Core Components Specification 1.8

document id

Version 0.4

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

April 28, 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.

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.

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.*
>
> 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

---

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

[2] CCTS lines 2162-2163

Specify that the name of a property (i.e. Property Term) should reflect the role played by that property's content *relative to* the Aggregate Core Component in which that property is 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]

Knowing its property allows us to identify (name) a Core Component (either a Basic or subsidiary Aggregate Core Component) within the Aggregate Core Component that contains it. That is, this property's relationship to the other elements of the CC meta-model.

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.

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.

---

[3] CCTS lines 2167-2168

[4] CCTS lines 1115-1116

A code is a character string used to replace a definitive value[5]. However, we need to understand the current practice for the use of codes within existing e-business vocabularies to remove some ambiguity.

Codes can be used in two different ways, either to define content (good) and or to define meta-data (to be avoided).

For example, ISO 3166.1 defines a set of valid Country Codes (i.e. a content code). So, if I wanted to unambiguously specify that the country of destination was Australia I would use the ISO 3166.1 code 'AU'.

However, EDIFACT 2005 (Date or time or period function code qualifier) defines the function of a date, i.e. it defines meta-data. These are the codes that tell you what kind of thing you are dealing with[6]. So, if I wanted to use a generic date field within an Order, I could qualify it by accompanying the date with a code of '4' meaning that the date is the 'date when an order is issued'. This is an alternative to defining the order date object explicitly. The same principle applies to the many qualifier and function codes used throughout EDI messages. This concept goes to extremes with things like EDIFACT 1131 (Code List Identification code), defining meta-meta-data – an interoperability nightmare. These meta-data codes do not represent Core Components, although they do provide clues to properties or perhaps contexts for Core Components. They should not be used as Core Component codes. They are an attempt to convey semantic meaning via coded references rather than explicitly within the library itself and are therefore unnecessary within the ebXML Core Component library.

Therefore, our comparison focuses on codes as used for defining data content.

If a code is used as a way of representing the value of something, an identifier provides unambiguous identification of an object[7]. The two are not alternatives, they are complementary to each other.

They can be confused because we often use codes to identify things. Country codes uniquely identify countries. But we can also use strings of text (e.g. names such as URLs) and unbounded sets of values (e.g. unique numbers such as phone numbers, SSID, tax file numbers, etc.) as identifiers. The value of an identifier tells you which thing in a set of similar things you're talking about.[8]

---

[5] *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.* [CCTS]

[6] Gait Boxman –ebTWG list on 10th April 2002

[7] *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.* [CCTS]

[8] Gait Boxman –ebTWG list on 10th April 2002

In addition, codes are not always identifiers. It depends on the context of their use. For example, within the object class Country, the value 'AU' is the unique identification of Australia. But, when used in other contexts, such as Shipping Location, 'AU' will not be unique – it is still a code, but it is not an identifier of the Shipping Location.

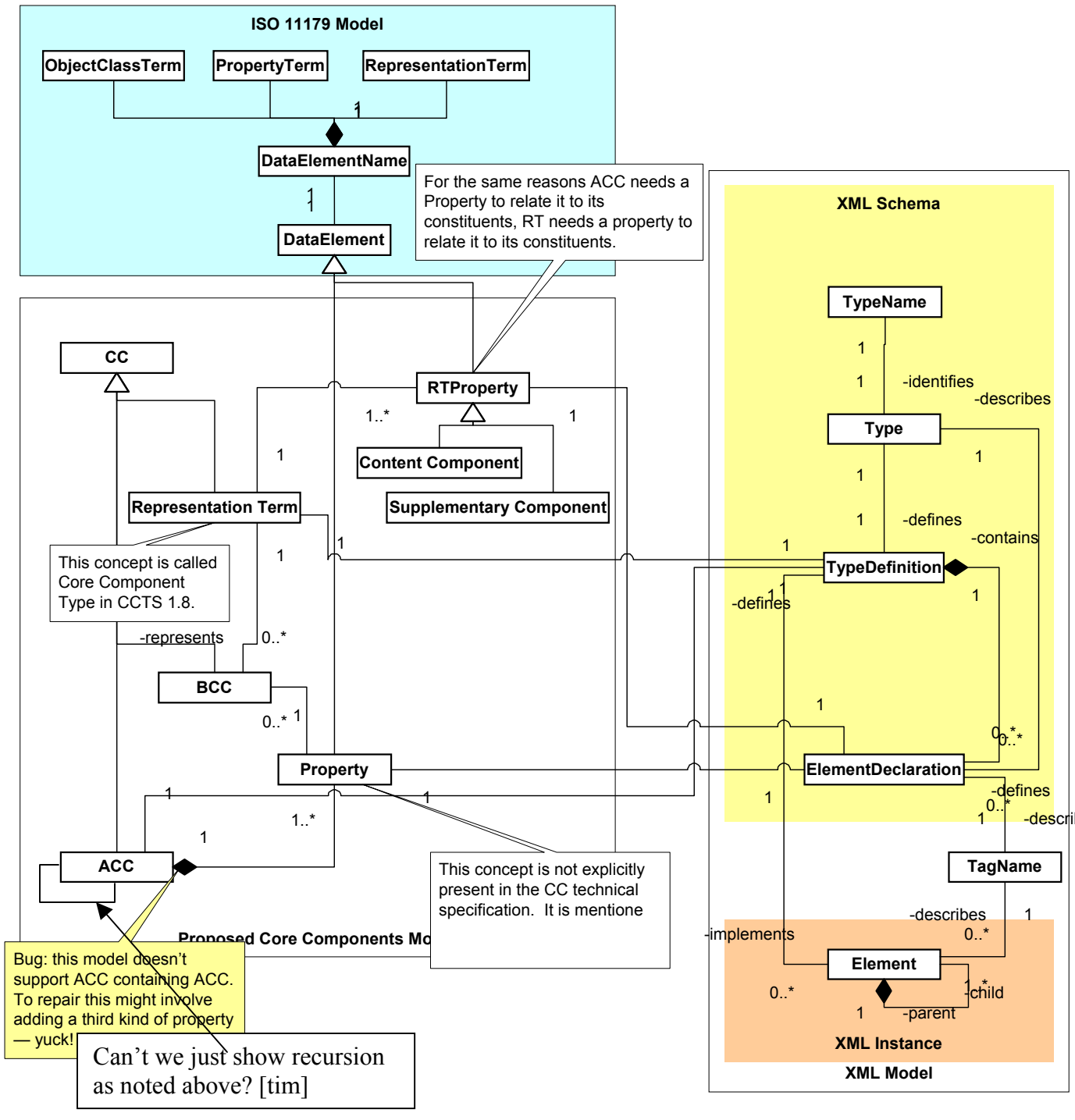
We have this mix of relationships because a code is a representation, whereas 'identification' is actually a property, a function of the use an object.

Proposal 9

> **The use of Identifier as a Representation Term (and Core Component Type if they still exist –[**Proposal 3**]) should be discontinued.**
>
> The property of 'Identifier' may exist as a Property Term. The representation of this may be as a Code, Text, Number, or any other Representation Term.

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:



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.	